# Genetic Search of Pickup and Delivery Problem Solutions for Self-driving Taxi Routing

Viacheslav Shalamov, Andrey Filchenkov, Anatoly Shalyto



# Genetic Search of Pickup and Delivery Problem Solutions for Self-Driving Taxi Routing

Viacheslav Shalamov, Andrey Filchenkov, and Anatoly Shalyto

ITMO University, Computer Technologies Lab,
49 Kronverksky Pr., 197101, St. Petersburg, Russia
shalamov@rain.ifmo.ru, afilchenkov@corp.ifmo.ru, shalyto@mail.ifmo.ru

**Abstract.** Self-driving cars belong to rapidly growing domain of cyber-physical systems with many open problems. In this paper, we study routing problem for taxies. In mathematical terms, it is well-known Pickup and Delivery problem (PDP). We use with the standard small-moves technique, which is to apply small changes to a solution for PDP in order to obtain a better one; and an approach that works with small-moves as mutations in genetic algorithms. We propose a strategy-based framework for managing set of small changes and suggest different strategies. We tested algorithms for routing on real-world dataset on taxi orders to airports in United Kingdom. The results show that algorithms using mixed strategies outperform algorithms using a single small move.

**Keywords:** self-driving car, autonomous car, routing, pickup and delivery, genetic algorithms, city taxi

## 1 Introduction

Usage of cyber-physical systems (CPS) such as medical devices, industrial robots or smart grid, showed extensive growth in the recent years [1]. It leads to high demand on algorithms for CPSs improving their performance and interaction with environment. Control systems of this type are thought to show high autonomy and to be able to solve typical tasks met in the domain of their application.

Self-driving cars (SDC), also known as autonomous cars, are one of the CPSs with rapidly growing importance: SDCs share market with ordinary (human-driving) vehicles, and potential effect of its introduction is often referred as a revolution [9, 21]. Consequences include improvements in traffic, vehicle usage and even implementation of new social strategy, so-called mobility as a service [25, 5]. List of organizations racing for SDC includes not only well-known vehicle manufacturing companies, but also Google, Uber [15] and Baidu [10].

One of the possible applications of SDC is a taxi service in airports. Airports are usually a very intensive hub for taxi routing, due to many people use taxis to get to an airport or get to a city after arrival. It explains why many airports all over the world have their own taxi services or at least taxi services accredited to be official for this airport. This policy shows several advantages. First of all, airport-associated taxis are less affected by the traffic situation in the city.

Second, tariffs for taxi are usually fixed or can be simpler adjusted to the current demand, since regular taxi services have to take into account all the demands for mobility across the city.

Many studies are devoted to SDC problem. They include:

- map generation and navigation [29, 19, 13];
- sensor system [12, 3];
- motion planning [20, 31, 17, 23, 2];
- distributed and parallel computational infrastructure [18, 24, 26];
- routing in different conditions [14, 6, 11, 32].

The last problems, namely routing, is actual for taxis. In its mathematical statement, this problem is known as Pickup and delivery problem (PDP) [16]. PDP is an optimization problem, relative to the most known travelling salesman problem (TSP). Both of them belong to the vehicle routing problem class, determining the optimal solution for problems of this class is known to be NP-hard [22]. The TSP can be considered as a case of the PDP, since one is required not only to visit each point, but also pickup items in several points and deliver them to other points [27]. A lot of constraints can be added to the general case of PDP [28]. The taxi routing problem may be understood as last-in-first-out single vehicle pickup and delivery problem (SVPDPL): 1) we assume that we need to build a route for a single vehicle; 2) it cannot pick another item before it had delivered current one [8]. SVPDPL is known to be NP-hard problem, thus, a number of heuristics for solving it are suggested.

In this paper, we use variable neighborhood search (VNS) approach [4] for solution search that constructs a new solution by applying local changes (small moves) to an already found one. We use the framework suggested in [30] involving usage of strategies.

The remainder of the paper is organized as follows. In Section 2, we define SVPDPL, its solution via local changes and genetic algorithms. In Section 3, we describe a concepts of strategy and different strategies we use in this paper. In Section 4, we describe experiment setups and the results of algorithm comparison. Section 5 concludes.

## 2 Problem statement and small moves

### 2.1 Pickup and delivery problem statement

We will follow the notation from [7] and [30]. The Pickup and delivery problem is represented with a weighted graph $G = (V, E, W)$, where $V$ is vertex set corresponding to the points, $E$ is edge set corresponding to the routes connecting points, and $W$ is a weight function that is defined on $V$ and $E$. It corresponds to loads on vertices and cost of travelling for edges. In this paper, we use $W$ representing the taxi routing problem:

- $W(v)$ equals 1 if $v$ is a vertex with a person to be picked up (*source*), and $-1$ if $v$ is a vertex, to which a person should be delivered (*destination*);

– $W(\{u, v\})$ is a distance between the points corresponding to $u$ and $v$.

Let $P$ denote the source set and $D$ denote the destination set, then $V$ is splited into pairs $(P, D) = ((p_1, d_1), \ldots, (p_n, d_n))$, $P, D \in V$.

We enumerate all the vertices: a source vertex $p_i$ receives index $i$, and the corresponding destination vertex $d_i$ receives index $i+n$. An edge $(i, j)$ can belong to $E$ if 1) $j = i+n$; or 2) $i > n$ and $j \leq n$, $i \neq n+j$. We need to find a Hamiltonian route, which minimizes its cost: $\sum_{j=1}^{2n-1} W(\{i_j, i_{j+1}\})$.

### 2.2 Small moves

Since the search of an exact solution is NP-hard, an heuristic known as "small moves" was suggested [4]. The main idea is to apply small changes to a found solution in order to obtain a new, better one. Today it is state-of-the art approach. In this work we use the following five types of the most commonly used small moves.

*Lin-2-Opt* substitutes two edges $(i_j, i_{j+1})$ and $(i_k, i_{k+1})$ with other two edges $(i_j, i_k)$ and $(i_{j+1}, i_{k+1})$, then it inverses the subpath $(i_{j+1}, \ldots, i_k)$. Our illustrated example is presented in web[1].

*Double-bridge*exchanges edges $(i_j, i_{j+1})$, $(i_k, i_{k+1})$, $(i_l, i_{l+1})$ and $(i_h, i_{h+1})$ with $(i_j, i_l + 1)$, $(i_k, i_{h+1})$, $(i_l, i_{j+1})$ and $(i_h, i_{k+1})$. Our illustrated example is presented in web[2].

*Couple-exchange* selects two orders $(i, n + i)$ and $(j, n + j)$, then it replaces $i$ with $j$ and $n + i$ with $n + j$. Our illustrated example is presented in web[3].

*Point-exchange* is analogous to the couple-exchange. The only difference is that not the pairs, but single points are exchanged.

*Relocate-block* (RB) inserts a sequence $(i_t, \ldots, i_{t+n})$ into the best possible place in the route. Our illustrated example is presented in web[4].

### 2.3 Applying genetic algorithms

The small moves described in the previous subsection can be easily understood as mutation operations, which can be applied to a solution in order to obtain a new, better one. This idea was realized in [30] for applying genetic algorithms for the solution search.

The application of genetic algorithms requires the following problem formalization: spices are solutions of the SVPDPL, and fitness function is the quality of such a solution (the length of the route, which should be minimized). Mutation operation is the application of a small move. Each generation consists of $K$ solutions, each solution gives a birth to $n$ new solution by allying a mutation operation, then some of them are chosen via the selection operation to form the next generation. Totally, there are $g$ generations.

[1] `http://genome.ifmo.ru/files/papers_files/AIAI2016/fig_L2O.pdf`
[2] `http://genome.ifmo.ru/files/papers_files/AIAI2016/fig_DB.pdf`
[3] `http://genome.ifmo.ru/files/papers_files/AIAI2016/fig_CE.pdf`
[4] `http://genome.ifmo.ru/files/papers_files/AIAI2016/fig_RB.pdf`

Let $N$ be equal to 50, which is the number of vertices in the graph. In [30], five different genetic algorithms were tested, we will use all of them.

1. $1 + 1$: each generation consists of one solution ($K = 1$), it gives a birth to one solution ($n = 1$), both the solutions are subjects of selection, the resulting solution forms the next generation ($E = 1$).
2. $1 + N$: $K = 1$, $n = \sqrt{N/2}$,, $E = 1$.
3. $1$, $N$: $K = 1$, $n = \sqrt{N/2}$, only the children are subjects of selection, $E = 1$.
4. $1 + N$ + Big mutation ($1+N+$BM): $K = 1$, $n = \sqrt{N/2}$, $E = 1$. In addition, every $g/4$ generations a Big mutations fires, during the which all the species are mutated $\sqrt(N)$ times.
5. $K+KN$: $K = \sqrt{N/4}$, $n = \sqrt{N/2}$, $E = K$.

## 3 Strategies

The question arises, which small moves can and should be used as mutation operations. The simplest way is to apply a small move of a certain type — this is a common practice. However, as it was shown in [30], applying different small moves is more beneficial. In that paper, one group of the genetic algorithms was allowed to pick mutation on each step randomly from the entire small move set (with the uniform distribution).

In this paper, we generalize such an approach with *strategy* concept. Assume we are given a set $\mathcal{M} = \{m_1, \ldots, m_M\}$ of small moves. Then let strategy $S = \{q_1, \ldots, q_M\}$ be a vector of probabilities, with which each of small move should be applied on the next step.

If we apply only a single small move, then this strategy is *pure*. In our case, we have 5 small moves: L20, DB, CE, PE, and RB. Thus, there five pure strategies: $S_{\text{L20}} = (1, 0, 0, 0, 0)$, $S_{\text{DB}} = (0, 1, 0, 0, 0)$, $S_{\text{CE}} = (0, 0, 1, 0, 0)$, $S_{\text{PE}} = (0, 0, 0, 1, 0)$, $S_{\text{RB}} = (0, 0, 0, 0, 1)$. However, as we have stated, mixed strategies will outperform pure strategies. The uniform, mixed strategy is: $S_{\text{mix}} = (1/5, 1/5, 1/5, 1/5, 1/5)$.

The second group of strategies is formed according to the following empirics: 1) CE and PE perform similarly, this is why we use only one of them; and 2) according to the results in [30], as well as this study, L2O seems to be the strongest algorithm. This is why it makes sense to consider strategies, which use L20 with combination of other small moves. Thus, we add five new strategies for comparison:

$$S_{\text{mix}-\text{PE}} = (1/4, 1/4, 0, 1/4, 1/4).$$

$$S_{\text{L20+DB}} = (1/2, 1/2, 0, 0, 0);$$

$$S_{\text{L20+CE}} = (1/2, 0, 0, 1/2, 0);$$

$$S_{\text{L20+RB}} = (1/2, 0, 0, 0, 1/2);$$

$$S_{\text{L20+mix}} = (1/2, 1/6, 0, 1/6, 1/6).$$

The last group of strategies we discuss in this paper are based on small-moves properties. We run experiments on 150 different subsets and collected

the following statistical evaluations: $n_j$ is the number of times, when $i$th small move is applied; $f_i$ is frequency of solution improvement by applying $i$th small move, and $c_i$ is frequency of solution improvement by applying $i$th small move ignoring cases, when it violates the problem constraints. Also let $n_i$ denote the computational complexity of the small move application. Thus:

$$S_{\mathrm{N}} \propto \{(1/n_i\} = (1,1,1,1,1/N); S_N = (0.249, 0.249, 0.249, 0, 249, 0.005),$$

$$S_{\mathrm{F}} \propto \{f_i\}; S_F = (0.327, 0.09, 0.158, 0.16, 0.265),$$

$$S_{\mathrm{C}} \propto \{c_i\}; S_C = (0.348, 0.166, 0.160, 0.16, 0.166),$$

$$S_{\mathrm{F/N}} \propto \{f_i/n_i\}; S_{F/N} = (0.4417, 0.1215, 0.2134, 0.2162, 0.007).$$

## 4 Experiments

### 4.1 Experiment setup and test data

VeeRoute company[5] kindly gave us dataset contacting information on 8,000 orders for taxies to transfer people from airports or to airports in the United Kingdom in March 2015. We used airport names and addresses to obtain geographical coordinates of the corresponding points using webservice mapquest[6]. Then we used the Mercator projection implementation library in library JMapProjLib[7] to transform them to 2D coordinates on a plane. Thus, we transformed the datasets into the dataset of two sets of points. Then we generated small datasets containing 50 random addresses as source and an airport as destination.

Each algorithm was run 10 times on 5 different subsets. Each experiment was held 10 times. We estimate *optimization ratio* (OR), which represents, how much we have decreased the length of the route in comparison with the original one:

$$\mathrm{OR} = \frac{d_0 - d_1}{d_0},$$

where $d_0$ is the length of the initial solution, and $d_1$ is the achieved length. We tested algorithms on $g = 12 \cdot N^2$ generations.

### 4.2 Strategies comparison

We compared all the strategies, presented in the previous Section, for each of genetic scheme, presented in Section 2. Results are presented in Table 1.

As we can see, 1+1 genetic scheme outperforms all the other genetic schemes in general. Strategies of the second type outperform corresponding strategies of the first type, with the only exception for the mixed strategy. However, statistic strategies show the best performance. Usage of small-moves complexity leads to

[5] `http://veeroute.com/`
[6] `www.mapquest.com/`
[7] `https://github.com/OSUCartography/JMapProjLib`

**Table 1.** Comparison of stretagies in OR.

| Strategy | 1+1 | 1+N | 1,N | 1+N,BM | K+KN |
|---|---|---|---|---|---|
| $S_{\mathrm{L20}}$ | 0.637 | 0.636 | 0.639 | 0.642 | 0.638 |
| $S_{\mathrm{DB}}$ | 0.410 | 0.406 | 0.437 | 0.416 | 0.391 |
| $S_{\mathrm{CE}}$ | 0.537 | 0.538 | 0.354 | 0.527 | 0.543 |
| $S_{\mathrm{PE}}$ | 0.576 | 0.572 | 0.397 | 0.573 | 0.580 |
| $S_{\mathrm{RB}}$ | 0.676 | 0.677 | 0.670 | 0.667 | 0.641 |
| $S_{\mathrm{mix}}$ | 0.729 | 0.726 | 0.695 | 0.715 | 0.700 |
| $S_{\mathrm{mix-CE}}$ | 0.716 | 0.704 | 0.703 | 0.708 | 0.693 |
| $S_{\mathrm{L20+DB}}$ | 0.647 | 0.639 | 0.655 | 0.633 | 0.606 |
| $S_{\mathrm{L20+PE}}$ | 0.679 | 0.683 | 0.622 | 0.674 | 0.686 |
| $S_{\mathrm{L20+RB}}$ | 0.702 | 0.698 | 0.701 | 0.697 | 0.674 |
| $S_{\mathrm{L20+mix}}$ | 0.722 | 0.717 | 0.726 | 0.704 | 0.702 |
| $S_{\mathrm{N}}$ | 0.736 | 0.705 | 0.631 | 0.709 | 0.679 |
| $S_{\mathrm{F}}$ | 0.726 | 0.716 | 0.712 | 0.717 | 0.701 |
| $S_{\mathrm{C}}$ | 0.733 | 0.717 | 0.696 | 0.731 | 0.704 |
| $S_{\mathrm{F/N}}$ | 0.736 | 0.710 | 0.648 | 0.710 | 0.678 |

the two best results. This is can simply be explained by the fact, that implying a small move (RB) consumes time $N$ greater, than any other small move, thus, applying such a normalization makes the mixed strategy more fair. However, it has no such a big improvement in other genetic schemes. When a more complex scheme is used, this difference becomes not so important. In these cases, statistic considerations have a higher impact. We can see, for the most of other schemes, $S_{\mathrm{F}}$, $S_{\mathrm{C}}$ are the best, due to the breed is bigger, than one, and then unsuccessful application of RB will not be so crustal for reaching the optima.

## 5 Conclusion and future work

In this paper, we have suggested a novel solution for self-driving car routing, which is based on applying strategies: small-moves with assigned probabilities, which changes a found solution. We have suggested and analyzed several strategies with several genetic schemes for generating them.

We have shown that small-moves ensembling is a very profitable step for results improvement. Also we have shown that the strategies that are based on the previous experience are very powerful. Another strong improvement was to connect probabilities of applying a small move with its computational complexity (time consumed).

We see several directions for future work. First, it may be useful to apply reinforcement learning in order to choose, which small move to apply next. Second, we plan to use meta-learning to predict the initial strategy and genetic algorithm scheme. Finally, it seems to be useful to choose the size of breed dynamically on each step (decreasing it), since strategies with big breed where good in the beginning of the breed.

**Acknowledgements.** Authors would like to thank Daniil Chivilikhin for useful comments. This work was financially supported by the Government of the Russian Federation, Grant 074-U01.

## References

1. Alur, R.: Principles of cyber-physical systems. MIT Press (2015)
2. Bender, P., Tas, O.S., Ziegler, J., Stiller, C.: The combinatorial aspect of motion planning: Maneuver variants in structured environments. In: Intelligent Vehicles Symposium (IV), 2015 IEEE. pp. 1386–1392. IEEE (2015)
3. Broggi[1], A., Bombini[1], L., Cattani[1], S., Cerri[1], P., Fedriga[1], R.: Sensing requirements for a 13,000 km intercontinental autonomous drive. In: Intelligent Vehicles Symposium (IV), 2015 IEEE. pp. 500–505. IEEE (2010)
4. Carrabs, F., Cordeau, J.F., Laporte, G.: Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading. INFORMS J. on Computing 19(4), 618–632 (2007)
5. Chong, Z., Qin, B., Bandyopadhyay, T., Wongpiromsarn, T., Rebsamen, B., Dai, P., Rankin, E., Ang Jr, M.H.: Autonomy for mobility on demand. In: Intelligent Autonomous Systems 12, pp. 671–682. Springer (2013)
6. Chu, K., Lee, M., Sunwoo, M.: Local path planning for off-road autonomous driving with avoidance of static obstacles. Intelligent Transportation Systems, IEEE Transactions on 13(4), 1599–1616 (2012)
7. Cordeau, J.F., Laporte, G., Ropke, S.: The Vehicle Routing Problem: Latest Advances and New Challenges, chap. Recent Models and Algorithms for One-to-One Pickup and Delivery Problems, pp. 327–357. Springer US (2008)
8. Cordeau, J., Iori, M., Laporte, G., Salazar Gonzlez, J.: A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with lifo loading. Networks 55, 46–59 (2010)
9. Dallegro, J.A.: How google's self-driving car will change everything. `http://www.investopedia.com/articles/investing/052014/how-googles-selfdriving-car-will-change-everything.asp` (2014), [Online; accessed 2016-02-15]
10. Davies, A.: Baidu's self-driving car has hit the road. `http://www.wired.com/2015/12/baidus-self-driving-car-has-hit-the-road/` (2014), [Online; accessed 2016-02-15]
11. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Path planning for autonomous vehicles in unknown semi-structured environments. The International Journal of Robotics Research 29(5), 485–501 (2010)
12. Ercan, Z., Sezer, V., Heceoglu, H., Dikilitas, C., Gokasan, M., Mugan, A., Bogosyan, S.: Multi-sensor data fusion of dcm based orientation estimation for land vehicles. In: Mechatronics (ICM), 2011 IEEE International Conference on. pp. 672–677. IEEE (2011)
13. Göhring, D., Wang, M., Schnürmacher, M., Ganjineh, T.: Radar/lidar sensor fusion for car-following on highways. In: Automation, Robotics and Applications (ICARA), 2011 5th International Conference on. pp. 407–412. IEEE (2011)
14. González, D., Perez, J., Lattarulo, R., Milanés, V., Nashashibi, F.: Continuous curvature planning with obstacle avoidance capabilities in urban scenarios. In: Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. pp. 1430–1435. IEEE (2014)

15. Hawkins, A.J.: Google vs. uber and the race to self-driving taxis. http://www.theverge.com/2015/12/16/10309960/google-vs-uber-competition-self-driving-cars (2015), [Online; accessed 2016-02-15]
16. Iori, M., Martello, S.: Routing problems with loading constraints. Top 18, 4–27 (2010)
17. Jafri, S.H., Kala, R.: Path planning of a mobile robot in outdoor terrain. In: Intelligent Systems Technologies and Applications, pp. 187–195. Springer (2016)
18. Jo, K., Kim, J., Kim, D., Jang, C., Sunwoo, M.: Development of autonomous carpart i: distributed system architecture and development process. Industrial Electronics, IEEE Transactions on 61(12), 7131–7140 (2014)
19. Jo, K., Sunwoo, M.: Generation of a precise roadway map for autonomous cars. Intelligent Transportation Systems, IEEE Transactions on 15(3), 925–937 (2014)
20. Kala, R., Warwick, K.: Planning autonomous vehicles in the absence of speed lanes using an elastic strip. Intelligent Transportation Systems, IEEE Transactions on 14(4), 1743–1752 (2013)
21. Kim, J., Kim, H., Lakshmanan, K., Rajkumar, R.R.: Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In: Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems. pp. 31–40. ACM (2013)
22. Laporte, G.: The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59(3), 345–358 (1992)
23. Li, X., Sun, Z., Kurt, A., Zhu, Q.: A sampling-based local trajectory planner for autonomous driving along a reference path. In: Intelligent Vehicles Symposium Proceedings, 2014 IEEE. pp. 376–381. IEEE (2014)
24. Martínez-Barberá, H., Herrero-Pérez, D.: Multilayer distributed intelligent control of an autonomous car. Transportation research part C: emerging technologies 39, 94–112 (2014)
25. Parc, C.F.: Mobility-as-a-service: Turning transportation into a software industry. http://venturebeat.com/2014/12/13/mobility-as-a-service-turning-transportation-into-a-software-industry/ (2014), [Online; accessed 2016-02-15]
26. Pérez, J., Milanés, V., Onieva, E.: Cascade architecture for lateral control in autonomous vehicles. Intelligent Transportation Systems, IEEE Transactions on 12(1), 73–82 (2011)
27. Savelsbergh, M.W., Sol, M.: The general pickup and delivery problem. Transportation science 29(1), 17–29 (1995)
28. Schneider, J., Kirkpatrick, S.: Stochastic optimization. Springer Science & Business Media (2007)
29. Schreiber, M., Hellmund, A.M., Stiller, C.: Multi-drive feature association for automated map generation using low-cost sensor data. In: Intelligent Vehicles Symposium (IV), 2015 IEEE. pp. 1140–1147. IEEE (2015)
30. Shalamov, V., Filchenkov, A., Chivilikhin, D.: Small-moves based mutation for pick-up and delivery problem. In: Proceedings of the Companion Publication of the 2016 on Genetic and Evolutionary Computation Conference. p. in press. ACM (2016)
31. Tanzmeister, G., Friedl, M., Wollherr, D., Buss, M.: Efficient evaluation of collisions and costs on grid maps for autonomous vehicle motion planning. Intelligent Transportation Systems, IEEE Transactions on 15(5), 2249–2260 (2014)
32. Valencia, R., Morta, M., Andrade-Cetto, J., Porta, J.M.: Planning reliable paths with pose slam. Robotics, IEEE Transactions on 29(4), 1050–1059 (2013)