An Initialized ACO for the VRPTW

Wei Shi and Thomas Weise

The USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI)
University of Science and Technology of China (USTC)
Hefei, Anhui, China

This is a preview version of the paper [1] (see page 9 for the reference).
Read the full piece at http://dx.doi.org/10.1007/978-3-642-41278-3_12.

Abstract. The Vehicle Routing Problem with Time Windows is an important task in logistic planning. The expenditure on employing labor force, i.e., drivers for vehicles, accounts for most of the costs in this domain. We propose an initialized Ant Colony approach, *IACO-VRPTW*, with the primary goal (f_1) to reduce the number of vehicle needed to serve the customers and the second-priority goal (f_2) of decreasing the travel distance. Compared with methods that optimize f_2, *IACO-VRPTW* can reach or reduce f_1 in 8 out of 18 instances of the Solomon benchmark set, at the cost of increasing travel distance slightly. *IACO-VRPTW* can effectively decrease the number of vehicles, travel distance and runtime compared with an ACO without initialization.

1 Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) is a common and important task in logistic planning. Here, the goal is to use capacity-restricted vehicles to serve several customers that require certain amounts of a product. Most related works try to minimize either first the number f_1 of these vehicles and then the distance these vehicles travel (f_2), or only focus on f_2.

In this paper, we present the results of a new Ant Colony Optimization (ACO) [2] approach – *IACO-VRPTW* – for solving VRPTWs. *IACO-VRPTW* can be distinguished from the related works by three main differences:

1. It does not follow a two-step approach, but optimizes both goals f_1 and f_2 all the time during its run (but *not* in a Pareto fashion).
2. It gives f_1 strict priority over f_2, i.e., a solution which requires fewer vehicles is better and amongst solutions with the same f_1-values, the one with the shortest travel distance f_2 is best. This approach is closer to practice, as costs for drivers are usually higher than costs for travel distance.
3. It applies an initialization procedure of the transition matrix in order to achieve better results earlier.

We will first outline the VRPTW in the next section and then discuss related works in Section 3. In Section 4 we describe our new approach. Experimental results are given in Section 5 and in Section 6, we conclude our paper and list our plans for future work.

2 Problem Definition

The Vehicle Routing Problems with Time Windows (VRPTW) is an extension of the VRP. Given are a set V of $m = |V|$ vehicles v_i, each having the same fixed capacity $k \in \mathbb{N}$ and being parked at the central depot c_0. Additionally, there is a set C with $n = |C|$ cities c_i to be serviced ($i \in 1..n$).

Each city c_i has a demand for a certain amount $w_i \in \mathbb{N}$ of a product and must be serviced by exactly one vehicle. The delivery of the product must begin between the earliest arrival time $e_i \in \mathbb{N}$ and latest arrival time $l_i \in \mathbb{N}$ of that city and takes the service time $s_i \in \mathbb{N}$. Vehicles cannot leave the depot before e_0 or arrive back at the depot after l_0.

The travelling cost is the distance $d_{i\,j}$ between two cities c_i and c_j. In the available benchmark cases, this usually is the Euclidean distance and city locations are points in the two-dimensional plane. The travel time is the distance divided by *speed*. In benchmark cases usually $speed = 1$.

A solution $r = (r_1, r_2, \ldots, r_m)$ of a VRPTW describes which city should be serviced by which vehicle. For each vehicle v_i with $i \in 1..m$, it provides an *ordered* set $r_i \subseteq 1..n$ describing the sequence of cities to visit. It is implied that vehicles start at the depot and return to it after finishing their schedule.

$$n = \sum_{i=1}^{m} |r_i| \tag{1}$$

$$\forall i, j : 1 \leq i \neq j \leq m \Rightarrow r_i \cap r_j = \emptyset \tag{2}$$

$$\text{feasible solution} \Rightarrow \forall i : 1 \leq i \leq m \Rightarrow k \geq \sum_{\forall c \in r_i} w_c \tag{3}$$

For any valid solution r, a set of conditions must hold, i.e., all customers must be serviced (Eq. 1), be serviced only once (Eq. $1 \wedge 2$), and no vehicle's capacity must be exceeded (Eq. 3).

Each vehicle v_i processes its tour r_i step by step from the beginning. The earliest service time $b_{r_{i,j}}$ for city $r_{i,j}$, i.e., the j^{th} city in schedule r_i (for vehicle v_i) is then given as Eq. 4. It is determined by the earliest arrival time $e_{r_{i,j}}$ of that city and the earliest service time $b_{r_{i,j-1}}$ of the city $r_{i,j-1}$ serviced before it, or the earliest departure time of the depot c_0, if it is the first city in schedule r_i ($j = 1$).

$$b_{r_{i,j}} = \begin{cases} \max\{e_{r_{i,j}},\ e_0 + t_{0\,r_{i,j}}\} & \text{if } j = 1 \\ \max\{e_{r_{i,j}},\ b_{r_{i,j-1}} + s_{r_{i,j-1}} + t_{r_{i,j-1}\,r_{i,j}}\} & \text{if } j > 1 \end{cases} \tag{4}$$

$$\forall i : 1 \leq i \leq n \Rightarrow e_i \leq b_i \leq l_i \tag{5}$$

Of course, in a feasible solution, all cities can be serviced within their respective time windows, as shown in Eq. 5.

Solving VRPTW generally has three goals: *(a)* use as few vehicles as possible (f_1, Eq. 6), *(b)* aim for the lowest travel cost (f_2, Eq. 7), and *(c)* minimize the vehicles' total waiting time. Yu et al. [3], Alvarenga et al. [4] choose the total travel distance f_2 as their objective. Homberger and Gehring [5] combined the

number of vehicles f_1 and f_2 as weighted sum in the objective function, while Li and Lim [6] include all three objectives as well as the schedule time. Common are also two-step processes [7, 8, 9], as we will outline in Section 3.

In the real world, using fewer vehicles will greatly decrease the transportation cost compared with more vehicles but slightly shorter distance. Thus, when we compare two solutions a and b, the one with fewer tours f_1 is considered as better. If $f_1(a) = f_1(b)$, then the one with the lower cost f_2 will win.

$$f_1(r) = |\{\forall i \in 1..m : |r_i| > 0\}| \tag{6}$$

$$f_2(r) = \sum_{i=1}^{m} \left[d_{0\, r_{i,1}} + d_{r_{i,|r_i|}\, 0} + \sum_{j=2}^{|r_i|} d_{r_{i,j-1}\, r_{i,j}} \right] \tag{7}$$

3 Related Work

Solomon [10] gives a detailed description of the VRPTW and the benchmark set used in most of the related works (and here as well). This set consists of six subsets: R1, C1, R2, C2, RC1, RC2. R2, C2, and RC2 are problems with a long scheduling horizon compared with R1, C1, and RC1, i.e., its trucks have a greater capacity and the time windows are broader.

As the VRPTW is NP-hard [10], meta-heuristics are used to approximately solve them. In order to satisfy the multiple objectives of VRPTW, Gambardella et al. [11] introduced a two-step approach, which then was used in many other works: Firstly they minimize the number of tours and secondly, under the given number of vehicles, minimize the total travel cost. Gambardella et al. [11] used the same metaheuristic for both steps. Berger and Barkaoui [7] adopted a Genetic Algorithm (GA) in both two phases. In the algorithm of Homberger and Gehring [9], the number of vehicles is minimized with an evolution strategy and the total distance by means of tabu search. Bent and Hentenryck [12] proposed a robust heuristic approach for VRPTW, based on simulated annealing and linear neighborhood spreads.

In our approach, both objectives are optimized at once. They are not combined to a weighted sum, but prioritized. We try to find schedules requiring fewer vehicles (f_1) and, at the same time, lower total travel cost (f_2).

We therefore use an ACO. The ACO idea has first been proposed in [2] and is inspired by the way how ants find paths based on pheromone released by other ants. In ACO, a solution is represented as pathes through a graph constructed by simulated ants. Like in nature, these ants start at a certain location and move forward, basing their decision of where to go on *(a)* pheromone τ (a dynamic value that can be changed) and *(b)* a sensed distance d to food (i.e., a static heuristic value). Ants that find good solutions may lay out additional pheromone $\Delta\tau$ that supports later ants to find short path.

Qi and Sun [8] proposed an ACO for VRPTW that optimizes f_1 and f_2 separately and outperforms the algorithm by Gambardella et al. [11]. Yu et al. [3] propose hybrid ACO algorithms that use local searches, such as tabu search

and neighborhood search for VRPTW. Instead of a local search, we use a special initialization procedure.

4 Approach: Initialized ACO

4.1 Preprocessing

The service beginning time b_i of c_i depends on the cities on the same tour visited before c_i. Thus, it is impossible to know the b_i of a city c_i until the cities visited before it have been chosen. However, we can rule out certain cities that would violate Eq. 5:

$$\text{can visit } c_j \text{ after } c_i \text{ in schedule } r_k \Rightarrow l_j \geq e_i + s_i + t_{i\,j} \tag{8}$$

Eq. 8 is one necessary condition for being able to reach c_j in time after servicing c_i. For each city c_i, we thus calculate the set of other cities that can be visited afterwards in the same schedule. We call this set the *domain* of c_i.

When searching for the next city to be visited, we just need to consider the cities in the current city's domain. This reduces the search space size and speeds up the optimization process.

4.2 ACO Strategy

In our algorithm, we use an elitist pheromone update strategy of type ANT-cycle [2]. Compared with ANT-quantity and ANT-density, ANT-cycle performs better as it updates pheromone globally not locally [2]. The updating rule for ANT-cycle is defined in Eq. 9, where $\Delta\tau$ denotes the pheromone change after one algorithm iteration:

$$\tau_{i\,j} = \rho\tau_{i\,j} + \Delta\tau_{i\,j} + \Delta\tau^{\star}_{i\,j} \tag{9}$$

$$\Delta\tau_{i\,j} = \sum_{k=1}^{a} \Delta\tau^{k}_{i\,j} \tag{10}$$

$$\Delta\tau^{k}_{i\,j} = \begin{cases} \frac{Q_1}{f_2^k} & \text{if ant } k \text{ travels from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$\Delta\tau^{\star}_{i\,j} = \begin{cases} \frac{Q_2}{d_{ij}m^{\star}} & \text{if the elitist ant travels from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Here, ρ is the evaporation coefficient that reduces the impact of the previous pheromone, a is the number of ants, Q_1 is the total quantity of pheromone an ant can leave on the way from c_i to c_j and f_2^k is the total travel distance of ant k. Eq. 11 is the normal pheromone update definition of ANT-cycle. $\Delta\tau^{\star}_{i\,j}$ in Eq. 12 is an extra prize for the elitist ant, i.e., the one that used the fewest vehicles (if there exists multiple such ants, then it is the one with the shortest travel distance). We modified the original formula of the update [2] to consider

both, the number of vehicles required of the elitist ant and the travel distance in Eq. 12.

The $(n+1) \times (n+1)$ pheromone matrix τ is used to calculate the transition probability matrix P according to Eq. 13, where $p_{i\,j}$ is the probability of an ant transferring from c_i to c_j.

$$P = \{p_{ij}\}, \quad p_{ij} = \begin{cases} \frac{(\tau_{i\,j})^{\alpha} * (1/d_{i\,j})^{\beta}}{\sum_j (\tau_{i\,j})^{\alpha} * (1/d_{i\,j})^{\beta}} & \text{if } c_j \text{ unvisited} \wedge \text{in domain of } c_i \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The index α determines the relative influence of the pheromone trails and β is the influence of the visibility. The probability $p_{i\,j}$ increases with higher pheromone density $\tau_{i\,j}$ and shorter distance $d_{i\,j}$.

In our algorithm, we have a certain number a of ants at the depot. Each of them must visit all the cities under the capacity and time window constraints. When an ant reaches its maximum capacity or has no next city to visit, it will automatically return to the depot. In the former case, its capacity is replenished and it begins the next tour.

4.3 Initialization Procedure

We use a simple directional selection to initialize the transition probability matrix P for the first ACO iteration. When we start the search, we set the transition probability from the depot to be higher for those cities which are close and have an earlier earliest start time. The top m cities with the lowest $h_{0\,j}$-values (Eq. 14) thus receive higher probabilities $p_{0\,i}$. The values γ_1, γ_2, δ_1, and δ_2 are weights.

$$h_{0\,i} = \gamma_1 d_{0\,i} + \delta_1 e_i \tag{14}$$

$$h_{i\,j} = \gamma_2 d_{i\,j} + \delta_2 d_{j\,0} \tag{15}$$

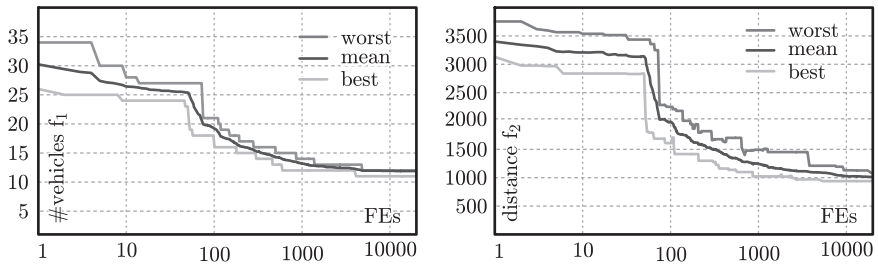
For ants leaving a city $i \neq 0$, we use the similar equation 15, which incorporates the distance back to the depot. The goal is to avoid that the ants end their tours in cities very far away from the depot. From the domain of each city c_i, we chose the top-num_1 cities according to Eq. 15. Amongst them, num_2 cities $num_2 < num_1$ are randomly chosen and receive higher probabilities.

5 Experiments

5.1 Experimental Setup

In our experiments, we use Solomon's benchmark for VRPTW [10]. Each complete solution r constructed by an ant will be counted as one function evaluation (FE). We grant 20 000 FEs to runs on 25 and 50 city problems and 300 000 FEs for 100 city problems. We use an Intel Core i3-3220 CPU with 3.30GHz.

The parameters of our algorithm have been introduced in Section 4. We set $Q_1 = 4000$, $Q_2 = 80$, $\rho = 0.9$, $num_2 = 8$, $\gamma_1 = 1$, $\delta_1 = 1$, $\gamma_2 = 1.5$ and



(a) Progress in terms of f_1. (b) Progress in terms of f_2.

Fig. 1: Convergence diagrams over all runs.

$\delta_2 = 0.5$. Then we test all combinations of $a \in \{10, 20, 30, 40, 50\}$, $\alpha \in \{0.5, 1\}$, and $\beta \in \{12, 3, 4\}$. For the 25-city problem, we set $num_1 \in \{10, 20\}$ and for the 50 and 100-city problems, we use $num_1 \in \{15, 25, 45\}$. We run the experiments with and without the initialization procedure discussed in Section 4.3, for 30 runs per setup.

5.2 Experimental Results

We compare the arithmetic means of the solution data over all configurations and find that $a = 50$, $\beta = 2$, and $\alpha = 1$ together with $num_1 = 20$, $num_1 = 25$, and $num_1 = 45$ for the 25, 50, and 100 city problems, respectively, perform best, i.e., lead to lower f_1 and f_2 values.

In Fig. 1, we illustrate the progress of our algorithm with that configuration for 100-city problem C101. The algorithm behaves similarly on the other instances. While f_1 decreases steadily, f_2 sometimes increases. This happens when new solutions with fewer vehicles appear that initially have a longer travel distance.

We now compare our results obtained with the above settings with the solutions from [3, 13] that were obtained by highly specialized algorithms. From Table 1, we see that the mean number $\overline{f_1}$ of vehicles of 8 out of 18 problem types are equal to or smaller than in [3, 13]. As our primary goal is to find solutions with fewer vehicles and then minimize the total travel distance, the results show that *IACO-VRPTW* is effective. It performs particularly well in problems with longer scheduling horizon (C2, RC, and RC2). Our method has some problems with the larger-scale instances. It can be seen that *IACO-VRPTW* finds good solutions in short time (less than 150s) for 100-city problems. This is probably due to the very limited number of FEs granted to the experiments, but we needed that setting for these first experiments to identify good configurations.

In Table 2, we compare our algorithm (*IACO-VRPTW*) to an algorithm version without initialization (*ACO-VRPTW*). The results of type C1, R1, and RC1 with 25 cities are aggregated under point 25_1, those of C2, R2, and RC2 under

Table 1: Comparision of *IACO-VRPTW* with the results published in [3, 13] in terms of mean objective values ($\overline{f_1}$, $\overline{f_2}$) and mean runtime $\overline{\mathrm{RT}}$.

	From [3, 13]		*IACO-VRPTW*		relative error ϵ		Runtime
Instance	$\overline{f_1}$	$\overline{f_2}$	$\overline{f_1}$	$\overline{f_2}$	ϵf_1	ϵf_2	$\overline{\mathrm{RT}}$ in s
25_C1	3.00	190.59	3.00	195.27	**0.00%**	2.46%	0.87
25_R1	4.92	462.87	4.92	490.15	**0.00%**	5.89%	0.96
25_RC1	3.25	350.24	3.50	374.01	7.69%	6.79%	0.95
25_C2	2.00	214.46	1.63	221.34	**-18.75%**	3.11%	0.81
25_R2	2.73	382.17	1.64	434.12	**-39.92%**	13.59%	0.91
25_RC2	2.88	319.53	1.88	370.99	**-34.78%**	16.11%	0.86
50_C1	5.00	361.69	5.22	400.22	4.40%	10.66%	2.45
50_R1	7.75	766.13	8.25	878.89	6.45%	14.71%	3.60
50_RC1	6.50	729.24	7.38	837.84	13.46%	14.89%	2.63
50_C2	2.75	357.50	2.38	397.71	**-13.64%**	11.25%	2.32
50_R2	4.11	634.03	2.50	782.35	**-39.19%**	23.39%	2.64
50_RC2	4.29	585.24	3.71	841.90	**-13.33%**	43.86%	2.73
100_C1	10.00	826.70	10.78	1074.38	7.78%	29.96%	115.61
100_R1	12.75	1155.89	16	1506.22	25.49%	30.31%	130.39
100_RC1	12.38	1342.42	15.63	1719.98	26.26%	28.13%	122.35
100_C2	3.00	587.36	3.88	788.92	29.17%	34.31%	143.55
100_R2	3.73	906.28	4.45	1331.70	19.51%	46.94%	144.89
100_RC2	4.25	1049.57	5.38	1613.93	26.47%	53.77%	126.79

25_2, and so on. From the table, it is obvious that the initialization procedure improves the result quality significantly and even the runtime.

6 Conclusions

In this paper, we proposed an initialized ACO – *IACO-VRPTW* – to solve vehicle routing problems with time windows. Our primary goal was to reduce the number of vehicles. We therefore adopt an elitist ANT-cycle model that awards ants finding schedules with fewer of vehicles, which may also accelerate the convergence speed.

In our experiments, we find that *IACO-VRPTW* can reach or reduce the vehicle numbers in 8 out of 18 problem types, at the cost of increasing the travel distance slightly. For practical considerations, however, that small cost increase is far out-weighted by reduced personnel expenses. We find that our simple directional selection-based initialization procedure is effective in decreasing the vehicle number, total cost, and runtime.

In the near future, we will incorporate local search methods into our algorithm in order to refine the solutions found by the ants.

Table 2: Comparison of our method with initialization (*IACO-VRPTW*) to the approach without (*ACO-VRPTW*) in terms of mean objective values ($\overline{f_1}$, $\overline{f_2}$) and mean runtime $\overline{\text{RT}}$

	ACO-VRPTW			*IACO-VRPTW*			relative error ϵ		
Type	$\overline{f_1}$	$\overline{f_2}$	$\overline{\text{RT}}$/s	$\overline{f_1}$	$\overline{f_2}$	$\overline{\text{RT}}$/s	ϵf_1	ϵf_2	$\epsilon\overline{\text{RT}}$
25_1	4.29	396.47	1.08	4.27	396.76	0.93	**-0.57%**	0.07%	**-15.39%**
25_2	2.12	372.42	1.10	2.11	371.79	0.87	**-0.53%**	**-0.17%**	**-27.34%**
50_1	7.79	782.76	2.91	7.82	785.33	2.73	0.40%	0.33%	**-6.74%**
50_2	3.19	715.50	2.84	3.17	720.32	2.58	**-0.47%**	0.70%	**-10.31%**
100_1	15.15	1500.26	9.36	15.13	1495.02	8.68	**-0.14%**	**-0.35%**	**-7.91%**
100_2	5.59	1260.77	9.70	5.57	1257.13	9.15	**-0.53%**	**-0.29%**	**-6.00%**

Bibliography

[1] Wei Shi and Thomas Weise. An Initialized ACO for the VRPTW. In Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao, editors, *Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'13)*, volume 8206/2013 of *Lecture Notes in Computer Science (LNCS)*, pages 93–100, Hefei, Anhui, China: Empark Grand Hotel, October 20–23, 2013. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-642-41278-3_12.

[2] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In *European Conference on Artificial Life*, pages 134–142, Paris, France, December 11–13, 1991.

[3] B. Yu, Z.Z. Yang, and B.Z. Yao. A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications*, 38(1):435–441, 2011. doi: 10.1016/j.eswa.2010.06.082.

[4] G.B. Alvarenga, G.R. Mateus, and G. de Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computer and Operations Research*, 34(6):1561–1584, 2007.

[5] Jörg Homberger and Hermann Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR*, 37:297–318, 1999.

[6] Haibing Li and Andrew Lim. Local search with annealing-like restarts to solve the vrptw. *European Journal of Operational Research*, 150(1):115–127, 2003.

[7] Jean Berger and Mohamed Barkaoui. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research*, 31(12):2037–2053, 2004.

[8] Chengming Qi and Yunchuan Sun. An improved ant colony algorithm for vrptw. *2008 International Conference on Computer Science and Software Engineering*, pages 455–458, 2008.

[9] Jörg Homberger and Hermann Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

[10] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[11] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. Technical Report IDSIA-06-99, IDSIA, Lugano, Switzerland, 1999.
[12] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4): 515–530, November 2004. doi: 10.1287/trsc.1030.0049.
[13] Marius M. Solomon. Vrptw benchmark problems. URL `http://web.cba.neu.edu/~msolomon/problems.htm`.


```
@inproceedings{SW2013AIAFTV,
  author    = {Wei Shi and Thomas Weise},
  title     = {{An Initialized ACO for the VRPTW}},
  booktitle = {Proceedings of the 14th International Conference on
               Intelligent Data Engineering and Automated Learning
               (IDEAL'13)},
  editor    = {Hujun Yin and Ke Tang and Yang Gao and Frank Klawonn
               and Minho Lee and Thomas Weise and Bin Li and Xin Yao},
  publisher = {Berlin, Germany: Springer-Verlag GmbH},
  address   = {Hefei, Anhui, China: Empark Grand Hotel},
  series    = {{Lecture Notes in Computer Science (LNCS)}},
  volume    = {8206/2013},
  chapter   = {12},
  pages     = {93--100},
  year      = {2013},
  month     = oct # {~20--23, },
  doi       = {10.1007/978-3-642-41278-3_12},
  sciids    = {BJS17},
  sciwos    = {WOS:000329908900012},
  inspec    = {13955885},
},
```