Solving the Dial-a-Ride Problem Using an Adapted Genetic Algorithm

Stjepan Zelić[1], Marko Đurasević[2], Domagoj Jakobović[2], and Lucija Planinić[2]

[1] Hypefy World, Zagreb, Croatia
[2] University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia
zellich93@gmail.com, marko.durasevic@fer.hr, domagoj.jakobovic@fer.hr, lucija.planinic@fer.hr

Abstract. The dial-a-ride problem (DARP) deals with the transportation of people from source to destination locations. One of the most common use cases is in the transportation of elderly or sick people, and as such it represents an important problem to consider. Since DARP is NP-hard, it most often has to be solved using various heuristic methods. Previous studies demonstrated that metaheuristics are suitable for solving this kind of problem. However, in most cases, basic metaheuristics have been considered without any adaptation to the problem, which could potentially limit their performance. Therefore, in this study a GA is proposed and several of its elements adapted for solving DARP. The obtained results show that the proposed algorithm can achieve better results than similar methods from previous studies. Moreover, the experiments demonstrate that the results can be improved by considering some constraints as soft constraints and including them in the cost function to give the algorithm more flexibility in the search.

Keywords: Genetic algorithm · Dial a ride problem · Optimisation.

1 Introduction

The dial-a-ride problem (DARP) is a special form of the vehicle routing problem (VRP) that involves the transportation of people rather than goods. In DARP, users make requests to be picked up from a specific location at a specific time and taken to another location by a specific time. The goal of the problem is to schedule a fleet of vehicles to meet the user's needs as much as possible, but also to minimise the duration of the route. DARP has many practical applications in the real world, including door-to-door transportation of elderly or disabled people [5], cab services [13], emergency services [13], and demand-responsive mass transit [11]. Since DARP is a special case of VRP, it also belongs to the category of NP-hard problems. Therefore, there is no known algorithm that provides optimal solutions in a reasonable amount of time. Most of the time, one has to resort to metaheuristics that have proven their strength in many areas such as scheduling [16], cryptography [14], rostering [2], transportation [1], and similar.

DARP has already received considerable attention in the literature. One of the first studies dealing with DARP, in which a sequential insertion heuristic is proposed, was done by Jaw et al. [8]. The problem was also addressed in [10] using simulated annealing. Tabu search (TS) was applied in [4] to a problem where travel time must be minimised by considering all user requests. A genetic algorithm (GA) for DARP was proposed in [9] that solves the problem of [4]. The main difference between these works is that several strict constraints are modelled as cost functions that are optimised, which gives some flexibility to GA. Another GA was used in [6], in which the authors test different algorithm configurations. An overview of different DARP models and solution methods can be found in [5]. An extension of DARP that allows users to change vehicles during their trip is solved in [12] using an adaptive large neighbourhood search algorithm. A hyperheuristic approach to solving DARP is proposed in [15]. This method finds the best heuristic strategy for applying simple operators that can be applied to new problems. In [11], an online version of DARP was considered where the optimisation routine runs continuously during system execution. A parallel extension of the TS method for DARP was proposed in [13]. A variant of the problem, where different trip types are studied, is investigated in [7]. In [3], the authors consider a flexible DARP variant in which only a portion of the user requests are predetermined.

The above overview shows that this problem is still intensively researched and many new DARP variants are proposed and investigated. In this paper we consider the original DARP variant described in [4] and [9]. The problem is solved using an adapted GA that incorporates some domain-specific information in its evolutionary process by adapting the solution initialisation procedure and the applied crossover operator. The goal of this research is to gain initial insights that can be used in subsequent studies to further improve the results and that can also be applied to solve the extended DARP variants.

The rest of the paper is organised as follows: Section 2 gives an introduction to DARP. The GA adapted for DARP is described in Section 3. The experimental setup and the results obtained by the proposed GA are described in Section 4. Finally, the conclusion of the paper and future research directions are outlined in Section 5.

2 Dial-a-ride problem

The DARP under consideration is modelled based on the problem defined in [4, 9]. In this problem, there are n customer requests for transportation, given as a list of $2n$ locations. Each request has a pickup location, denoted with item i in the list, and a delivery location (item $n + i$). The locations are modelled as a fully connected graph in which a travel distance d_{ij} is defined between all locations i and j. For each location, there is a time window $[b_i, e_i]$ that defines the service of the request at that location, either for pickup or delivery. Ideally, the service at locations should only occur within these time windows. A service time s_i required at each location is also defined. Each customer request has a

specific number of places that the user takes in the vehicle, which are taken at the pickup location and released at the delivery location. Customers also specify a maximum amount of time they would like to spend in the vehicle. To meet user requests, a fleet of m vehicles is available. Each vehicle k starts at the depot location D and returns there after completing all requests. Each vehicle has a constant capacity of C and a maximum route duration. Since it is assumed that all vehicles are identical, both values are the same for all vehicles.

Usually several objectives are considered in DARP, out of which a single cost function is defined as a weighted linear combination of the individual cost functions. The cost functions considered in this study are:

- f_1 - total route duration - the total duration of the routes for all vehicles
- f_2 - total ride time - the total time that the customers spent riding in the vehicles
- f_3 - total wait time - the time that the vehicles spent idle while waiting to service a request
- f_4 - total late time - the total time that the vehicle was late, meaning that it arrived at a location after its defined time window
- f_5 - total amount of ride time violation - the excess amount of time that the customer spend driving in the vehicle above their requested ride time
- f_6 - total maximum route violation - the excess amount of time that the cars spent driving over their given maximum route duration

In [4] only the functions $f_1 - f_3$ were minimised, while the remaining functions were not used because they were modelled as hard constraints (i.e., no lateness was allowed). However, in [9], the authors modelled some constraints as cost functions, which allowed them to obtain better results. The total cost function to be minimised is defined as $f = w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3 + w_4 \cdot f_4 w_5 \cdot f_5 + w_6 \cdot f_6$. The weights $w_1, \ldots, w_6$ can be freely chosen to determine the significance of each cost function. The magnitudes of the functions f_1 and f_2 are usually similar, while f_3 is usually one order of magnitude smaller. However, the weights have been set as $w_1 = w_2 = w_3 = 1$, since initial experiments have shown that the algorithm nevertheless focuses quite well on optimising the cost function f_3 with such a setting. The cost functions f_4 and f_5 were usually about 5 times smaller than f_1 and f_2, while the cost function f_6 was usually equal to 0. Therefore, their weights were set to $w_4 = w_5 = w_6 = 5$ to focus equally on the cost functions modelling more stricter requirements such as lateness.

3 Genetic algorithm for DARP

To find solutions to the considered DARP problem, a GA is adapted for it. The solutions are represented by two chromosomes, an integer and a permutation chromosome. The integer chromosome specifies which vehicle each user request is associated with. The permutation chromosome represents the order in which customer requests are processed. Figure 1 represents an example of a problem with 2 vehicles and 5 requests. Since there are 10 requests, this means that

values 1-5 in the permutation vector represent pickup requests, while values 6-10 represent delivery requests. In this example, vehicle 0 will first handle the pickup of request 4 and then immediately handle its delivery (request 9). Then the vehicle will handle the second request. Vehicle 1, on the other hand, will first process three pickup requests (requests 2, 3, and 1) and then perform their delivery (requests 8, 7, and 6). The order of the delivery requests does not have to be the same as the order of the pickup requests. This is also clear in the example, because delivery request 7, which corresponds to the pickup request 2, is handled after the delivery request 8, which corresponds to the pickup request 3.

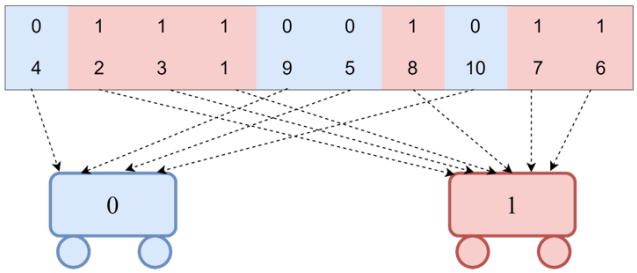

Fig. 1. Solution representation used by the GA

Instead of generating the initial population completely at random, a simple heuristic initialisation was used to construct the initial solutions. The outline of this procedure is shown in Algorithm 1. First, a permutation of requests is randomly generated by ensuring that each pickup request appears before the corresponding delivery request in the solution. Then, the following process is repeated until all requests are served. If the first request in the list is a pickup request, the list of vehicles with free space is first determined. If no such vehicles are available, the request is placed at the end of the list until certain delivery requests are processed and the vehicles free up space for new requests. If there are vehicles with free space, a priority p_i is calculated for each vehicle i and the current request j as

$$p_i = \left| \frac{e_j + b_j}{2} - (t_i + d_{ij}) \right|,$$

where e_j and b_j represent the end and the beginning of the time window, t_i is the current time of the vehicle, and d_ij is the distance between the vehicle and the pickup request location (the time it takes to reach a location is equal to the distance). This priority indicates how close to the middle of the time window the vehicle would arrive. The vehicle with the lowest value is then selected to serve the request. This solution initialisation method has demonstrated to achieve better results in preliminary experiments than if the initial population is generated completely randomly. On the other hand, if the first request in the list

is a delivery request, a check is made to see if the corresponding pickup request has already been served. If yes, this delivery request is assigned to the corresponding vehicle. Otherwise, the request is placed back in the list of requests to be considered when its corresponding pickup request is handled.

Algorithm 1 Initial solution construction procedure

requests ← create random permutation of requests
while *requests* not empty **do**
    currentRequest ← first request from *requests*
    if *currentRequest* is a pickup request **then**
        fVehicles ← currently free vehicles
        if *fVehicles* is empty **then**
            Place *currentRequest* at the end of the *requests* list
        else
            for each vehicle v_i in *fVehicles* **do**
                Calculate $p_i = \left| \frac{e_j + b_j}{2} - (t_i + d_{ij}) \right|$, where j denotes the location of *currentReqest*
            end for
            selectedVehicle ← select the vehicle with the lowest p_i
            Assign *currentRequest* to *selectedVehicle*
        end if
    else
        if the corresponding pickup request of *currentRequest* is already assigned to a vehicle **then**
            Assign *currentRequest* to the vehicle which contains its corresponding pickup request
        else
            Place *currentRequest* at the end of the *requests* list
        end if
    end if
end while

For the crossover operator an adapted PMX crossover, denoted as partially car mapped crossover (PCMX), is used. Unlike in the PMX crossover, in which two random crossover points are selected, in this variant a number of vehicles are selected and then all the genes associated to those vehicles are copied to the child individual. The remaining genes are then filled in a similar way as it is done in the original PMX crossover by copying over those requests form the second parent which are not yet present in the child individual. Figure 2 shows an example of the crossover performed on a solution for a problem with 10 requests and 2 vehicles. In this case the route for vehicle 1 is copied over from the first parent to the child individual. The requests which belong to the other vehicles are then filled from the second parent. If a request already exists in the child, then it would be mapped in the same way as in PMX to a request which

does not exist in the child individual, and that request would be copied to the child.

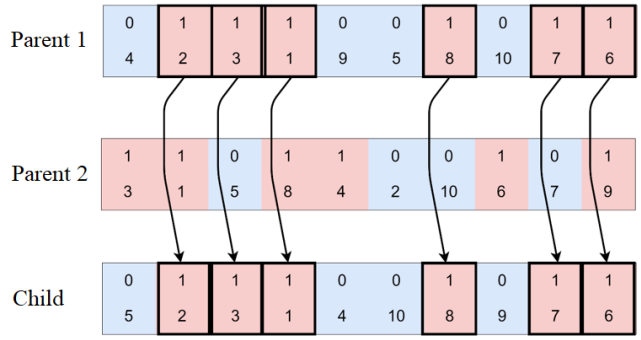

Fig. 2. Example of the PCMX crossover

The mutation is performed by simply swapping the order of the requests for two customers. Two customers are randomly selected (they can be assigned to the same or different vehicles) and then their pickup and delivery requests are swapped. An example of this mutation is shown in Figure 3. In this example it can be seen that the pickup requests 4 and 2 are swapped, as well as their corresponding delivery requests 9 and 7.

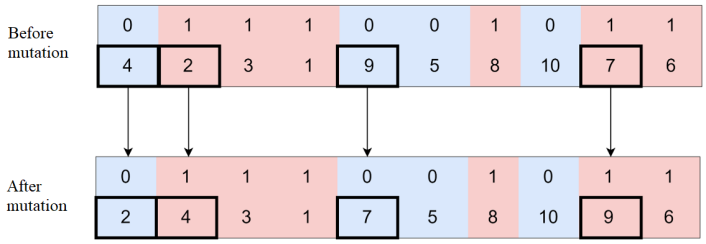

Fig. 3. Example of the swap mutation

It is possible that during evolution a certain number of constraints are not satisfied. Therefore, after each modification of an individual, a procedure is used to check the validity of the solutions and make a correction. First, it is checked whether all delivery requests appear after their respective pickup requests. If this is not the case, the two requests are simply swapped. Second, it checks if the vehicle capacity constraint is satisfied at all points in time. This is determined by finding the point where the capacity constraint is violated in the solution and then moving a delivery request ahead of that point to make room in the

vehicle. Using these corrections ensures that the algorithm only works with valid solutions throughout the evolution process.

4 Experimental study

4.1 Benchmark setup

The experimental study will be conducted on the dataset which is proposed in [4]. This dataset consists out of 20 problem instances which contain between 24 and 144 customers, and between 3 and 13 vehicles. The instances are divided into two groups, instances from R1a to R10a were generated with narrow time windows, whereas instances from R1b to R10b have been generated with wide time windows. For each instance the GA was executed 10 times. The parameters of the GA were fine tuned in preliminary experiments. A population size of 200 individuals, mutation probability of 0.1, the 5-tournament selection for selecting individuals, and stopping criterion of 1500 generations were used.

The results will be directly compared to the results obtained by previous studies from Cordeau and Laporte [4] and Jorgensen et al. [9]. The results obtained in the previous two studies are summarised in Table 1. It should be outlined that in these studies not all instances were considered, therefore only a subset of instances is denoted in the table.

Table 1. Overview of the results from the literature

	Cordeau and Laporte [4]			Jorgensen et al. [9]		
Instances	Route duration	Waiting time	Ride time	Route duration	Waiting time	Ride time
R1a	1041	252	477	881	211	1095
R2a	1969	470	1367	1985	724	1977
R3a	2779	292	3081	2579	607	3587
R5a	4250	500	5099	3870	833	6154
R9a	3597	94	6251	3155	323	5622
R10a	5006	315	8413	4480	721	7164
R1b	907	143	630	965	321	1042
R2b	1719	198	1214	1565	309	2393
R5b	4296	552	4615	3596	606	6105
R6b	5309	630	6134	4072	449	7347
R7b	1299	102	990	1097	129	1762
R9b	3679	147	5362	3249	487	5581
R10b	4733	113	7969	4041	362	7072
Total	40584	3808	51600	35537	6082	56900

4.2 Results

The results obtained by the proposed GA are shown in Table 2. The table outlines the three main objectives considered in previous studies: route duration, ride time, and waiting time. "Avg." denotes the average of 10 executions obtained for these objectives, while "Best" denotes the value for the objective obtained by

the solution with the best fitness. Also, the average values for the late times and ride time violations per customer are given to illustrate how much the obtained solutions violate these constraints. The results in the tables are marked with '†' if they are only better than the results of Cordeau and Laporte, with '*' if they are better than those of Jorgensen et al., with '+' if they are better than the results of both studies, and with '-' if they are worse than the results of both studies. Note that the results for the instances that were not solved in previous studies are not marked. The last row shows the aggregated results for the instances that were also used in the previous studies to make the cumulative results comparable.

Table 2. Overview of the obtained results

Instance	Route Duration		Waiting Time		Ride Time		Late Time	Ride Time Violation
	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Avg.
R1a	890*	972*	114+	201+	1138-	694†	0.49	0.07
R2a	1601+	1975+	164+	491+	2190-	1969†	0.72	1.60
R3a	2353+	2387+	117+	93+	3487†	2958†	0.34	0.92
R4a	3252	3598	270	548	4635	4495	0.53	1.20
R5a	3813*	3958*	193+	317+	5885†	4790+	0.98	1.32
R6a	4691	4773	279	323	7228	7133	0.92	2.58
R7a	1273	1354	133	162	1571	1295	1.50	0.19
R8a	2271	2254	46	14	3349	2803	0.89	2.86
R9a	3225*	3305*	64†	118†	5835*	5947*	4.72	2.52
R10a	4422*	4518*	97+	102+	8099*	7796*	5.19	4.26
R1b	788*	766*	31+	4+	984†	667†	0.48	0.06
R2b	1499*	1422*	55+	6+	2108†	1733†	0.42	1.32
R3b	2306	2282	69	34	3370	2555	0.34	0.42
R4b	3001	2941	75	38	4353	3636	0.20	0.82
R5b	3749*	3981*	135+	242+	5618†	5130†	0.58	1.34
R6b	4492*	4456*	149+	139+	6653†	6171+	1.13	1.01
R7b	1150*	1120*	19+	10+	1571†	1358†	0.54	1.12
R8b	2329	2355	100	88	3505	2658	0.96	2.17
R9b	3287*	3337*	45+	90+	5962-	5415†	1.88	2.98
R10b	4388*	4442*	66+	70+	7734*	7084*	3.62	2.00
Total	35659*	36637*	1249+	1882+	57264-	51711†	21.08	20.52

The results show that the proposed algorithm can achieve some improvements over the results obtained in the studies of Cordeau and Laporte and Jorgensen et al. For the waiting time cost, the proposed algorithm always obtained better results than both methods. For the route duration cost, the algorithm always obtained better results than the method of Jorgensen et al. Finally, for the ride time cost, better results were obtained for multiple instances than in the study of Cordeau and Laporte, but the overall results obtained for this criterion were worse than in both studies. It should be mentioned that compared to the results of Cordeau and Laporte, the route duration and travel time costs obtained by the proposed method are similar (within a range of 1%). However, we obtained a much smaller value for vehicle waiting time, by a factor of 5. This improvement was possible due to the flexibility provided by treating some constraints as soft

constraints (late time). Jorgensen et al. also treated late times as soft constraints. Unfortunately, these values are not reported in the paper and it is not possible to determine the extent to which these constraints were not met. However, the proposed GA was able to achieve better results for the route duration and wait time objectives.

Although the problem was solved by treating late times as soft constraints, it can be noted that the violations of late times and ride times are not very extensive. The average of late times is usually not greater than one minute, and the maximum late time was 5 minutes for instance R10a. On the other hand, the ride time violations were also usually in the range of one or two minutes. Such small violations of constraints should not cause much user dissatisfaction, but should give the algorithm more flexibility in finding better solutions for other criteria. Therefore, it seems to be more beneficial to treat such constraints as soft and optimise them together with the other objectives, as this seems to have a positive effect on the other objectives.

5 Conclusion

The obtained results show that with the initial adjustment of the GA it is possible to improve the results for DARP. For one of the considered criteria the algorithm achieved a significant improvement over the existing results, while the results for the other two criteria were mostly consistent with those of the other studies. This performance was achieved by including more problem-specific elements in the algorithm, but also by allowing some constraints not to be met. Since in this study the problem was addressed only briefly and with only a few adjustments to the algorithm, there is still much room to improve the results through further adjustments and fine-tuning for the problem under consideration.

In future studies, the goal is to test the proposed method on other data sets used in related surveys. It is also intended to adapt the proposed approach to cover other DARP variants not considered in this work. A more thorough study with different metaheuristic algorithms will be conducted to propose alternative and more efficient methods for DARP. Since DARP usually considers multiple objectives simultaneously, another obvious research direction would be to apply multi-objective algorithms.

Acknowledgements

This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333.

References

1. Baker, B.M., Ayechew, M.: A genetic algorithm for the vehicle routing problem. Computers & Operations Research **30**(5), 787–800

(2003). https://doi.org/https://doi.org/10.1016/S0305-0548(02)00051-5, https://www.sciencedirect.com/science/article/pii/S0305054802000515
2. Burke, E.K., Curtois, T., Post, G., Qu, R., Veltman, B.: A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research **188**(2), 330–341 (2008). https://doi.org/https://doi.org/10.1016/j.ejor.2007.04.030, https://www.sciencedirect.com/science/article/pii/S0377221707004390
3. Busing, C., Comis, M., Rauh, F.: The dial-a-ride problem in primary care with flexible scheduling (2021)
4. Cordeau, J.F., Laporte, G.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological **37**(6), 579–594 (2003). https://doi.org/https://doi.org/10.1016/S0191-2615(02)00045-0, https://www.sciencedirect.com/science/article/pii/S0191261502000450
5. Cordeau, J.F., Laporte, G.: The dial-a-ride problem (darp): Models and algorithms. Annals OR **153**, 29–46 (06 2007). https://doi.org/10.1007/s10479-007-0170-8
6. Cubillos, C., Rodriguez, N., Crawford, B.: A study on genetic algorithms for the darp problem. pp. 498–507 (06 2007). https://doi.org/10.1007/978-3-540-73053-8_50
7. Dong, X., Rey, D., Waller, S.T.: Dial-a-ride problem with users' accept/reject decisions based on service utilities. Transportation Research Record **2674**(10), 55–67 (2020). https://doi.org/10.1177/0361198120940307, https://doi.org/10.1177/0361198120940307
8. Jaw, J.J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.: A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological **20**(3), 243–257 (1986). https://doi.org/https://doi.org/10.1016/0191-2615(86)90020-2, https://www.sciencedirect.com/science/article/pii/0191261586900202
9. Jorgensen, R., Larsen, J., Bergvinsdottir, K.: Solving the dial-a-ride problem using genetic algorithms. Journal of the Operational Research Society **58** (10 2007). https://doi.org/10.1057/palgrave.jors.2602287
10. JR., J.W.B., KAKIVAYA, G.K.R., STONE, J.R.: Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. Engineering Optimization **30**(2), 91–123 (1998). https://doi.org/10.1080/03052159808941240, https://doi.org/10.1080/03052159808941240
11. Lois, A., Ziliaskopoulos, A.: Online algorithm for dynamic dial a ride problem and its metrics. Transportation Research Procedia **24**, 377–384 (2017). https://doi.org/https://doi.org/10.1016/j.trpro.2017.05.097, https://www.sciencedirect.com/science/article/pii/S2352146517303782, 3rd Conference on Sustainable Urban Mobility, 3rd CSUM 2016, 26 – 27 May 2016, Volos, Greece
12. Masson, R., Lehuédé, F., Péton, O.: The dial-a-ride problem with transfers. Computers & Operations Research **41**, 12–23 (2014). https://doi.org/https://doi.org/10.1016/j.cor.2013.07.020, https://www.sciencedirect.com/science/article/pii/S0305054813001998
13. Pandi, R.R., Ho, S.G., Nagavarapu, S.C., Tripathy, T., Dauwels, J.: Gpu-accelerated tabu search algorithm for dial-a-ride problem. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp. 2519–2524 (2018). https://doi.org/10.1109/ITSC.2018.8569472
14. Picek, S., Jakobovic, D., Miller, J.F., Batina, L., Cupic, M.: Cryptographic boolean functions: One output, many design criteria. Applied Soft Computing

40, 635–653 (2016). https://doi.org/https://doi.org/10.1016/j.asoc.2015.10.066, https://www.sciencedirect.com/science/article/pii/S1568494615007103

15. Urra, E., Cubillos, C., Cabrera-Paniagua, D.: A hyperheuristic for the dial-a-ride problem with time windows **2015**, 1–12 (2015). https://doi.org/10.1155/2015/707056, https://doi.org/10.1155/2015/707056
16. Vlašić, I., Đurasević, M., Jakobović, D.: A comparative study of solution representations for the unrelated machines environment. Computers & Operations Research **123**, 105005 (2020). https://doi.org/https://doi.org/10.1016/j.cor.2020.105005, https://www.sciencedirect.com/science/article/pii/S0305054820301222