Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 2. pp. 209–218
doi: 10.14794/ICAI.9.2014.2.209

The basic problem of vehicle scheduling can be solved by maximum bipartite matching*

Viktor S. Árgilán, János Balogh, Attila Tóth

Institute of Applied Sciences
Gyula Juhász Faculty of Education
University of Szeged
{gilan,balogh,attila}@jgypk.u-szeged.hu

Abstract

In the operative planning of public transport, the vehicle scheduling problem (VSP) is one of the most important tasks. In the single depot vehicle scheduling problem (SDVSP) there is only one depot, each vehicle being in the same depot; and the task is to construct a valid set of vehicle schedules in such a way that each timetabled trip is covered by a vehicle schedule (vehicle shift). The objective to be minimized is the sum of the cost of the timetabled trips and the cost of the trips without passengers, where the latter trips are the deadhead trips and the pull-in and pull-out trips. (The vehicles have to return to the depot at the end of the day.) This optimization problem can be solved by solving a minimum perfect matching problem of a weighted bipartite graph (Bertossi et al., Networks, Vol. 17, 1987). Here, we consider the basic problem of vehicle scheduling (BVSP), which is a special (fleet minimization) case of SDVSP, where the cost to be minimized is just the number of vehicles used (vehicle schedules) and we do not consider any other possible costs. We show that BVSP can be solved by using the maximum matching of a non-complete, unweighted bipartite graph.

Keywords: public transport, Single Depot Vehicle Scheduling Problem, Basic Vehicle Scheduling Problem, fleet minimization, bipartite graph matching

MSC: 90B06, 90B10, 90C27

1. Introduction

The Vehicle Scheduling Problem (VSP) is a classical optimization problem of public transport. For VSP, we are given a set of timetabled trips for a given day. The

*This study was partly supported by the European Union and the European Social Fund through project "Supercomputer, the national virtual lab" grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0010 and the Chinese-Hungarian bilateral project TET-12-CN-1-2012-0028.

timetable is given in advance. (In general, the timetable is prescribed by the local council.) The other tasks of the vehicles are determined by the public transport company staff during the operative planning phase. With VSP, we need to construct the scheduling of the vehicles for a given day in such a way that it provides a feasible and efficient scheduling for each timetabled trip. Each timetabled trip is given by its departure and arrival time, departure and arrival stations as geographical places. Deadhead trips are also allowed, among the appropriate stations of two compatible trips, and also between the depot and any station. A pair of two timetabled trips is compatible if there is sufficient time for a deadhead trip between them. The timetabled trips and the deadhead trips should have (non-negative) costs. Solving VSP involves scheduling a fleet of vehicles to cover a set of tasks at minimum total cost, where the total cost is the sum of each cost.

In the single depot vehicle scheduling problem (SDVSP), there is only one depot and each vehicle starts from this depot at the beginning of the day and returns to this depot at the end of the day. The SDVSP problem was solved by Bertossi et al. in 1987 [2]. These authors showed that SDVSP is equivalent to finding a minimum perfect matching of a complete, weighted bipartite graph. This proof is elegant and quite nice.

In the multiple depot vehicle scheduling problem (MDVSP) [3], there are more depots and for each trip a subset of the depots is described. Only the vehicles from that depot set can supply vehicles for the trips. In this case, in a valid schedule each pair of consecutive trips must be compatible and each trip of a vehicle shift can be supplied from the depot where the given vehicle originates from. Each cost (the cost of any timetabled trip or deadhead trip) may depend on the depot of a vehicle. In [2], Bertossi et al. showed that MDVSP is an NP-hard problem. In this article, we will focus on SDVSP, but a general review of both the single and multiple depot cases can be found in [1,3,4,6,7].

Here, we consider the so-called basic problem of vehicle scheduling [5], or BVSP for short. BVSP is a special case of SDVSP, where the cost in SDVSP is just the number of vehicles in the schedule (i.e. the number of different vehicle shifts in the solution). In other words, we are only interested in determining the minimum number of vehicles (each vehicle is assumed to be of the same type), which is necessary for meeting the (daily) scheduling requirements. BVSP can be interpreted as a special fleet minimization problem as well. We will show that BVSP can be solved by finding a maximum matching of a non-complete, unweighted bipartite graph. The number of the edges of this graph depends on the number of possible deadhead trips. Usually, the number of elements of the set of deadhead trips – which turns out to be the same as the number of the edges in our bipartite graph – depends on the type of the public transport, but typically this has around $n^2/2$ number of edges (e.g. for a dense bus timetable of a city).

2. Definitions, background

Below, we just focus on BVSP, which is a special case of SDVSP. The task of BVSP is to determine the minimum number of vehicles needed to supply the trips of a prescribed timetable in a given time period (typically one day) based only on the timetabled trips of this interval and the possible deadhead trips. The time period is typically a day, but other time intervals can be considered as well. We will suppose that we have a sufficient number of vehicles, that each vehicle is of the same type, and that from the depot the departure of any trip is realizable in time. (A vehicle can run to any departure time and geographical station of any trip in time.) Apart from the number of the vehicles, we will not consider any other cost. We will not consider the cost of the trips or the cost of the depot trips (pull-in and pull-out trips).

Let us denote by T the set of timetabled trips in the given time period. Each $i \in T$ timetabled trip is given by its departure time *dt(i)*, departure station *dg(i)*, arrival time *at(i)*, and arrival station *ag(i)*. Two trips $i \in T$ and $j \in T$ are compatible if the same vehicle can supply them, i.e. a deadhead trip is feasible between i and j. That is, if $at(i) \leq dt(j)$ and the length *dt(j)-at(i)* is less than or equal to the travel time of a deadhead trip from *dg(i)* to *ag(i)*. (We shall assume that each vehicle requires the same travel time to perform the same trip.)

In BVSP, we need to determine the minimum number of vehicles required based on the data of the timetabled trips (*at(i), dt(i), ag(i), dg(i)*, for each $i \in T$) and deadhead trips (compatible i,j pairs of trips). Of course, this number is a lower bound for the number of vehicles of any valid vehicle scheduling in a given time interval.

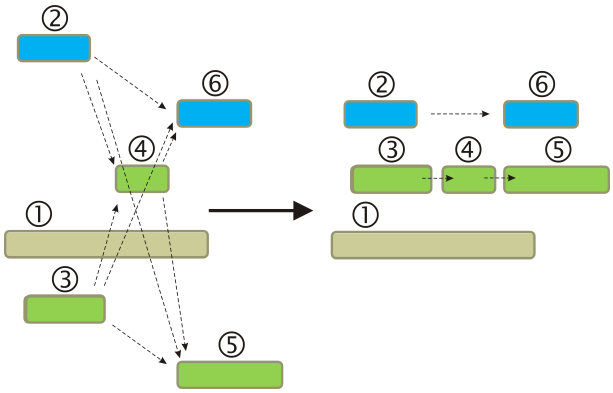

Figure 1: An example of six trips that can be solved using three vehicles. Here the geographical stations are not shown, just the possible deadhead trips

Figures 1 and 2 provide pictorial examples. In both figures the set of trips is the same, but the set of deadhead trips is different; and for this reason the solutions of

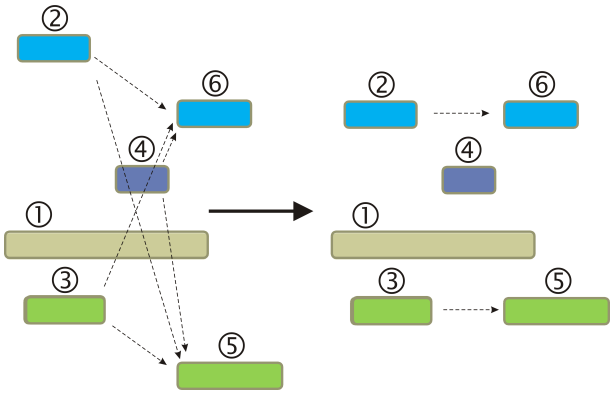

Figure 2: A modified version of the example in Figure 1, which shows that by allowing a different set of deadhead trips, a similar case can be solved by just using four vehicles

the examples may be also different.

The approach of Bertossi et al. [2] for finding the solution of SDVSP can be applied to find the solution of BVSP as well. It solves the problem by treating it as a minimum cost matching problem of a complete bipartite graph. In the case of BVSP, the weight assigned to each edge is 1, representing of the appropriate deadhead trips or depot trips.

In the following, we will describe our solution to BVSP. We get the solution by solving the maximum matching problem of a non-complete, unweighted bipartite graph. This graph is built by using the timetabled trips and by using the deadhead trips between any compatible pair of trips. The nodes of the graph represent the end-points of the timetabled trips (which are the end-points of the deadhead trips as well).

With BVSP, the cost we have to consider is 1 for each vehicle used. The cost of a valid scheduling is the sum of these, i.e. the number of vehicles used. In a valid scheduling each consecutive pair of timetable trips is compatible with each other in each chain (shift) of a vehicle, and we have to cover each timetabled trip by exactly one vehicle (shift). With BSVP we do not have to consider any other costs of the timetabled trips, the costs of the deadhead-trips or the costs of the depot trips. We will assume that in theory each vehicle is of the same type; more precisely we will suppose that there is a homogeneous vehicle fleet with a sufficient number of vehicles and that we do not have any special vehicle requirements for the trips. Each vehicle can perform each trip with the same journey time. Furthermore, in the BVSP model we will not consider the driver shifts, regulations for the working time and driving time, or any other requirements for the driver shifts (meal breaks, etc.). Actually, in the BVSP model we can assume that the drivers are robots that do not need any breaks, and so on.

Hence, we are interested in determining the minimum number of vehicles (and

the same number of drivers) that is necessary to provide each timetabled trips. From the above reasons, it can be readily seen that it will be a theoretical lower bound for it.

3. The solution of BVSP

Next, we will show how we can determine the minimum number of vehicles needed to supply the trips for a given time period, i.e. to get a valid vehicle scheduling in the period. This time interval I can be chosen arbitrarily. It is clear that when choosing a full day for this interval we will get the solution for a given day as well. This choice is typical, but our method also works for arbitrarily longer or shorter time periods.

Here, we identify all the five possibilities for the relationship between the time interval I and a timetabled trip i. There are the following:

A) *d(i)* is in interval I and *a(i)* is not,
B) *a(t)* is in interval I and *d(i)* is not,
C) both of *a(i)* and *d(i)* are in *I,*
D) neither *a(i)* nor *d(i)* is in *I*, but the trip includes the whole interval *I*,
E) neither *a(i)* nor *d(i)* is in *I*, and the whole the trip i is outside of *I*.

Below, we will restrict ourselves to the trips of classes A, B, C and D. Thus we will only consider those trips where the intersection of the time-period of the trip and the time-interval I is not empty. We will not consider the trips of E, because they will not matter if we analyze the interval I.

Let us denote by n_1, n_2, n_3, and n_4 the number of trips in classes A, B, C, and D, respectively. It is clear that using the number of vehicles $n_1+n_2+n_3+n_4$ the problem can be solved. (It is a feasible solution, where each trip is a vehicle schedule itself.) Otherwise, we cannot use fewer than $n_4$number of vehicles in I (because for each trip of class D we need a different vehicle). But it is possible to decrease the sum $n_1+n_2+n_3$, if we can create chains from the trips of class *A, B*, and *C*. A chain must be a valid schedule shift, so the consecutive timetabled trips of the chains must be compatible ones. Figure 3 shows an example for trips of the classes A, B, C, and D of a time-interval I_k.

We should add that if we consider a whole day as time-interval I, i.e. the timetable of a day, then each trip is (generally) in class C. Otherwise, if the given time interval I is a sub-interval of a day, and the length of I is less than the length of the shortest timetabled trip of a day; then we will not have trips in class *C*, independently of the choice of *I*. The matching of the trips will then produce chains of at most 2 trips.

In BVSP, we are interested in finding a valid vehicle schedule that employs a minimum number of vehicles. For this, we have to chain (concatenate) some trips of I and for an optimal solution we must do it in an optimal way. If interval I is a day, then the chains will be the (vehicle) shifts of that day, and all the vehicle

shifts together is in theory a vehicle schedule of the day. This is why BVSP may be viewed as a fleet minimization version of SDVSP.

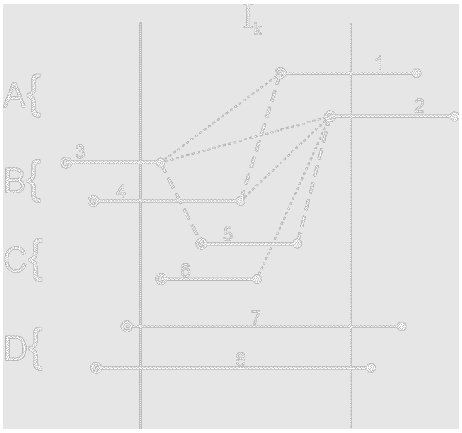

Figure 3: An example for the trip-classes A, B, C, and D in a time interval I_k. The horizontal axis represents the time (we do not represent the geographical places of the stations, only the possible deadhead trips among the timetabled trips). Here, $n_1=n_2=n_3=n_4=2$. We give a feasible solution with 5 vehicles, with 3 deadhead trips, so there will be a matching with value $m=3$ in the underlying bipartite graph. Hence, with $n_1=n_2=n_3=n_4-m=5$, five vehicles are sufficient to solve the problem. (We have to apply different vehicles to supply the trip chains 4-1, 3-5-2, and the trips 6, 7, and 8 themselves as special 1-length-shifts.)

For an interval I, we construct the bipartite graph $G_I=(U_I,V_I,E_I)$, where U_I and V_I are the nodes of the two parts of the graph, and E_I is the set of the edges. We assign it to the timetabled trips of the interval I; then for each trip of classes B and C we create a node in U_I, representing the departure time of the appropriate trips and U_I contains only these n_2+n_3 nodes. For each trip of classes A and C, we create a node in V_I. Each node of V_I represents the arrival time of exactly one trip of A or C. And V_I does not contain other node, so V_I has n_2+n_3 number of nodes. The edge set E_I of G_I is defined in such a way that there is an edge between two $u\in U_I$ and $v\in V_I$ nodes if and only if the trips represented by u and v are compatible; i.e. they can be performed in a consecutive way (there is sufficient time for a deadhead trip between them). Figure 4 shows the (automatically generated) bipartite graph for a small example.

After these preliminaries, we will consider a matching of the bipartite graph G_I, containing exactly m number of edges, and $e=(u,v)$ is an edge of the matching. Then the two trips, represented by u and v can be concatenated into one chain (of a valid vehicle scheduling) because of the definition of the edges of G_I. Here, the appropriate trips are compatible.

Based on the above points, we can build an iterative procedure which constructs

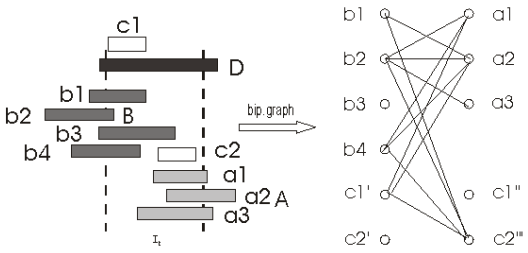

Figure 4: A small example for the generation of the bipartite graph

the chains of consecutive timetabled trips by transforming each edge of the matching of G_I to a deadhead trip of a valid chain. For each step of the procedure, we consider an edge of the matching that we have not examined in the previous steps. This procedure has m number of steps, because the matching has m edges. Before the first step of the procedure, we have $n_1+n_2+n_3+n_4$trips and each can be treated as a special, 1-length-chain of trips. Each iterative step involves the examination of an edge of the matching. Doing this, we concatenate two chains of the previous chains, based on the deadhead trip represented by the edge currently been examined in the matching, and during a step the number of vehicles of the previous step decreases by one. Then after the last (m-th) step of the procedure, we find that $n_1+n_2+n_3+n_4$-m vehicles are sufficient to cover all the trips of I.

A scheduling obtained in this way is valid, because each trip is covered by a vehicle (each trip is in a chain) and the consecutive trips are compatible because of the edges of G_I and the chains are independent (each trip is covered by only one), since the appropriate edges of G_I are disjoint due to the definition of matching. We should add that this procedure works in both directions. Now, the following lemma holds:

Lemma 3.1. *Let us suppose that G_I is a bipartite graph constructed in the above way, representing the trips (of the classes A B, and C) of a time-interval I. In this case there is a matching of G_I that contains m number of edges, if and only if there is a valid scheduling for the timetabled trips of I, using $n_1+n_2+n_3+n_4$-m vehicles. (BVSP assigned to I can be solved using $n_1+n_2+n_3+n_4$-m vehicles.)*

Proof. We will prove this using mathematical induction with i, where i is the number of edges in a matching of G_I.
Base case (case i=0). If we have an empty matching that contains no edge, then there is no deadhead trip in the scheduling. It is a valid scheduling that uses $n_1+n_2+n_3+n_4$ number of vehicles, where each vehicle chain contains exactly one trip. Thus we have $n_1+n_2+n_3+n_4$ independent chains of trips. The converse holds as well. That is, if the number of chains is $n_1+n_2+n_3+n_4$ in a chain, then there is no deadhead trip between any pair of timetabled trips.
Inductive step (the step from i to i+1). Let us suppose that the above assertion

holds for i ($i \leq 0$), i.e. G_I has a matching of i edges if and only if there is a valid scheduling of the trips of i using $n_1+n_2+n_3+n_4$-$i$vehicles. That is, there are $n_1+n_2+n_3+n_4$-i disjoint(independent) chains of trips such that each chain can be supplied by one vehicle.

Let us consider a matching of G_I containing $i+1$ edges. If we consider an arbitrary edge of the matching, let us denote it by e' (and suppose that the two endpoints of e' are u' *and* v'). Then the other i edges of the matching identify $n_1+n_2+n_3+n_4-i$ disjoint ("independent") chains of trips based on the induction hypothesis; and this is a valid vehicle scheduling of the trips which can be served by $n_1+n_2+n_3+n_4-i$ vehicles. The two trips, represented by the two end-points of edge e, are included in two different chains of trips. (We note that the trip, represented by u is at the end of a chain, while the trip represented by v is at the beginning of a chain.) The two chains have no common trips because of the induction hypothesis (the chains were constructed based on i edges of the matching. Furthermore, we needed two different vehicles to serve these two chains.

Upon examining edge e', we see that it is possible to concatenate the two chains by a deadhead trip. The existence of edge $e'=(u',v')$ means that after the last trip of the first chain, whose trip is represented by u', can follow the trip represented by v' in a valid scheduling, after a possible deadhead trip (from the arrival station of the trips represented by u' to the departure station of the trip represented by v'). After concatenating the two chains, it is sufficient that only one vehicle supply the two chains instead of one, and the $n_1+n_2+n_3+n_4-(i+1)$ chains we get will be disjoint chains (because the $i+1$ edges of the appropriate matching of G_I are disjoint).

To prove that the converse also holds, let us suppose that we have a valid scheduling of the trips of I, using $n_1+n_2+n_3+n_4-(i+1)$ vehicles, covering each timetabled trip by exactly one vehicle. If we consider a deadhead trip d of a chain, and split this chain into two different chains, one chain is the part of the original chain before d, and the other chain is the part of the original chain after d. In this way, instead of one vehicle we need two vehicles to serve these two disjoint chains. (Such a deadhead trip exists, because $i+1>0$.) Then we get a valid scheduling by $n_1+n_2+n_3+n_4-i$ vehicles. Hence the scheduling we get is a valid one for the trips of I, using $n_1+n_2+n_3+n_4-i$ vehicles to cover them, and the same number of disjoint chains. Based on the induction hypothesis, there is a matching of G_I that contains i edges. Because of the definition of G_I, there is an edge $e_d$of E_I that represents the deadhead trip d. The two end-points of this edge are not covered yet by the matching, because the chains of the assigned scheduling are disjoint (independent) ones – each trip being covered exactly once. Then, adding the edge to the matching, we get a matching of the graph G_I that has m edges. (Here we used the fact that in the underlying bipartite graph G_I, the edges of the matching have no common points (what is the definition of a matching.) This proves the lemma. □

In this way, we proved that for each step of the iterative procedure we can decrease the number of vehicles needed by one.

Summarizing the above points, we see that we can indeed assign a matching of G_I to a feasible (valid) scheduling of the trips of I, and vice versa. To find the minimum number of vehicles, we need to determine the maximum matching of G_I. This connection is stated in the theorem below.

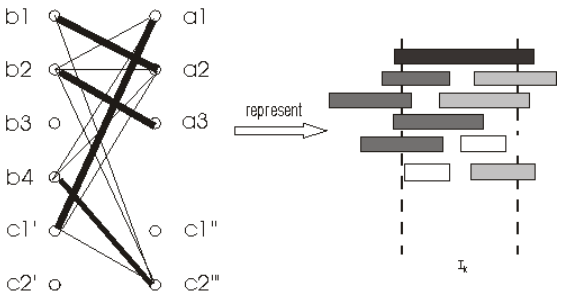

Figure 5: A small example: how we get a solution to BVSP from a solution to the maximum matching problem of the underlying bipartite graph (see the example in Figure 4)

Theorem 3.2. *Let G_I be a non-complete bipartite graph, constructed in the above-mentioned way for the timetabled trips, for a given time interval I. If the maximum matching of this graph G_I has m edges, then at least $n_1+n_2+n_3+n_4$-m vehicles are required in a valid vehicle scheduling to cover each timetabled trip in an interval I.*

Proof. Assume by contradiction that there is a valid scheduling of the trips of the time interval I, which uses fewer than $n_1+n_2+n_3+n_4$-m vehicles. We need m_4 vehicles for the trips of the class D, hence to serve the other trips (for the trips of the classes A, B, and C), we need $n_1+n_2+n_3$-m' vehicles, where $m'>m$. Let us consider such a valid scheduling of these trips of I. This valid scheduling contains $n_1+n_2+n_3-m'$ chains of trips, each one being assigned to exactly one vehicle. But based on Lemma 3.1, G_I has a matching with fewer than m edges, so a matching of the graph with m edges is not a maximum matching of the graph. In this way, we get a contradiction (arising from the fact that we assumed that using fewer than m vehicles we could get a valid vehicle scheduling for the trips). □

Figure 5 shows an example of the reconstruction of the solution of BVSP got from a matching.

Remark 3.3. If we assign non-negative costs to the timetabled and deadhead trips, then this method works by using a matching algorithm for the graph, where each edge has a cost, whose cost is the same as the cost of each deadhead trip. (The model dealt with above can be viewed as a special case of this, where each cost is 1.)

4. Summary

Here, we presented a solution for the basic problem of vehicle scheduling. We found its optimum via a solution of a maximum matching problem defined on a non-complete bipartite graph.

References

[1] ÁRGILÁN V., BALOGH, J., BÉKÉSI J., DÁVID B., GALAMBOS G., KRÉSZ M., TÓTH A., Ütemezési feladatok az autóbuszos közösségi közlekedés operatív tervezésében: Egy áttekintés, *Alkalmazott Matematikai Lapok*, (in Hungarian), 39 pages, to appear, 2014.

[2] BERTOSSI, A.A., CARRARESI, P., GALLO, G., On Some Matching Problems Arising in Vehicle Scheduling Models, *Networks*, Vol. 17 (1987), 271–281.

[3] BODIN, L., GOLDEN, B., ASSAD, A., BALL, M., Routing and Scheduling of Vehicles and Crews: The State of the Art, *Computers and Operations Research*, Vol. 10 (1983), 63–211, 1983.

[4] BUNTE, S., KLIEWER, N., An overview on vehicle scheduling models, *Journal of Public Transport*, 1(4), 299–317, 2009.

[5] DADUNA, J.R., Vehicle Scheduling, In Encyclopedia of Optimization, Second Edition, (eds. Floudas, C.A., Pardalos, P.M.), pp. 4027–4032, Springer, 2009. ISBN 978-0-387-74758-3

[6] DESAULNIERS, G., HICKMAN, M.D., Public Transit, In Handbook in OR & MS, (eds. C. Barnhart, C., Laporte, G.), Vol. 14, Chapter 2, Elsevier B.V., 2007.

[7] PEPIN, A.-S., DESAULNIERS, G., HERTZ, A., HUISMAN, D., Comparison of Heuristic Approaches for the Multiple Depot Vehicle Scheduling Problem, *Journal of Scheduling*, 12(1), 17–30, 2009.