Two Results on Car-sequencing Problems

Ian P. Gent*

Report APES-02-1998, April 1998†

Abstract

Car-sequencing problems have often been addressed using constraint programming techniques. I present solutions of two open problems in this domain. First I show that the problem is NP-complete. Then I show that a previously open benchmark problem proposed by Smith has no solution.

1 Introduction

The car sequencing problem is based on the problem of customising individual cars on a production line, and goes back in the constraint programming community to [1] having previously been defined by Parello, Kabat and Wos [5, 6]. More recently, the problem has again attracted attention as it provides fertile ground for the study of formulations and search methods [7, 8].

Each car requires some subset of the set of possible customising 'options'. We must schedule the order of cars to be put on the line. Unfortunately each option is subject to a capacity constraint. For example, if a car needs a sunroof, it may not be possible to install one on either of the next two cars. This would be a '1/3' constraint because of each three consecutive cars, at most one may have a sunroof. More generally, each option i has a capacity constraint p_i/q_i meaning that at most p_i cars in each consecutive sequence of q_i may have option i fitted. Note that this definition means that a 2/6 constraint is different to a 1/3 constraint. Each car requires a certain 'configuration' of options, and for simplicity we assume that these are all fitted simultaneously.

Car sequencing problems were introduced as a special case of job-shop scheduling problems. A special case of an NP-complete problem is not necessarily NP-complete, such as 2-SAT which is in P while SAT in general is NP-complete. Accordingly it has remained open whether the car-sequencing problem is in P or NP. By giving a reduction of the Hamiltonian Path problem into car-sequencing, I settle this question in the expected direction, that car-sequencing is indeed NP-complete.

The second contribution is the solution of an open benchmark problem proposed by Barbara Smith. While resisting intensive computational methods, I am able to give a simple demonstration that no acceptable sequence is possible.

2 NP-completeness

Car-sequencing is certainly NP-*easy*. To show that a given problem has a solution we only need to give the sequence of cars: this sequence can easily be checked for validity in polynomial time.

I will show that we can express any undirected Hamiltonian path (HP) problem as a car-sequencing problem. Solving an NP-hard problem by a polynomial reduction into car-sequencing shows that it in turn must be NP-hard. Because it is NP-hard and NP-easy it must be NP-complete.

*Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, United Kingdom. ipg@cs.strath.ac.uk

†APES reports are currently available from `http://www.cs.strath.ac.uk/~apes/apereports.html`

HP is the close relation of Hamiltonian circuit in which we seek a path through all nodes, visiting each exactly once and only traversing edges in a given graph: the difference from HC is that we start at a designated source and must end at a designated destination. Undirected Hamiltonian path is problem GT39 in Garey and Johnson [2].

2.1 Reduction of Hamiltonian Path to Car-sequencing

Hamiltonian Path is akin to car-sequencing in that both seek a permutation of a set of objects (nodes or cars) where the permutation must satisfy certain local constraints (edges existing in a graph or valid sequences of options.) My translation from from HP into car-sequencing will need three cars for each node in the graph, and I will call these 'node-cars', 'edges-cars' and 'vanilla cars'. So if there are N nodes there will be $3N-1$ cars (no vanilla car is necessary for the destination.) The options described below will force the cars for each node must come together in a group and in the following order

1. the first car can be seen as representing the node itself. Apart from the car representing the source node, this car will only be able to be sequenced immediately after a group of three cars representing a node which has an edge to this node.
2. the next car can be seen as representing the set of edges from the given node. This will be used to ensure that the next group of cars will represent a node accessible from this one.
3. finally there will come a vanilla car with no options, to serve as a buffer between the two cars above and the next group of three.

There will be $2N+4$ options. These form three groups: to control the basic ordering of cars; to tie together node-cars with their respective edges-cars; and to prevent consecutive groups of cars representing nodes not connected by any edge in the graph.

- **Control options.** There will be options called $C_1, \ldots, C_4$. These will be used to force cars to appear in order as node-edge-vanilla, starting with the source node-car and ending with the destination edges-car.
  - C_1 allows only 1 car in $3N-2$ to take it. This will force the node-car for the source to be the first car processed, and the edges-car for the destination to be the last.
  - C_2 allows 1 car in $3N-4$ to take it. This will force the edges-car for the source to be the second car processed and the node-car for the destination to be the penultimate.
  - C_3 allows 1 car in 3. This will force the node-cars to appear as every third car in the sequence, starting from the first.
  - C_4 also allows 1 car in 3 and will force the edges-cars similarly to appear as every third car, starting from the second.
- **Tying nodes and edges.** Because we have separate cars for a node and the edges out of that node, we must ensure that the correct edges-car follows each node-car. To force this there will be options $T_1, \ldots, T_n$. Each will allow only 1 car in 2.
- **Edges.** Finally we can control the path itself, which is after all the whole point. There will be options $E_1, \ldots, E_n$. These will allow only 1 car in 3. A node-car for node i will only need option E_i, while the edge car for node i will require option E_j for each node j for which there is *not* an edge ij in the original graph. This will prevent the node-car for node j appearing next.

I will now describe in detail what options each car requires. (Unless the following forces an option to be required, it will not be required.)

- Vanilla cars require no options at all.
- Node-cars
  - The node-car for the source requires option C_1.

- The node-car for the destination requires C_2.
- Each node-car including source and destination requires option C_3.
- The node-car for node i requires the two options T_i and E_i.

- Edge-cars
  - The edges-car for the source requires option C_2.
  - The edges-car for the destination requires option C_1.
  - Each edges-car including the source requires option C_4.
  - The edges-car for node i requires every option T_j *except* T_i which it does not require.
  - For $i \neq j$, the edges-car for node i requires option E_j if and only if there is *not* an edge from i to j in the original graph. The edges-car for node i does not require option E_i.

2.2 Example

It might help to see the translation in action. Figure 1 shows a graph with no Hamiltonian path from source node 1 to destination node 6.

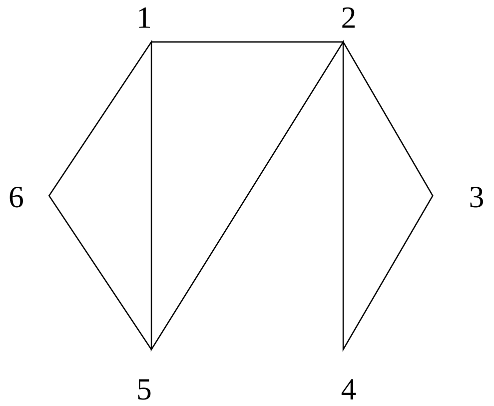

Figure 1: Example undirected graph with no Hamiltonian Path from 1 to 6. Addition of an edge between nodes 4 and 5 creates a path 1-2-3-4-5-6.

The translation will result in 17 cars in 13 different configurations. The only configuration with more than one car will be the vanilla configuration v. Each other car will correspond either to a node, and I will name the node-car for i as n_i; or to an edge, and I will name the edges-car for i as n_i. An x indicates that the configuration requires that option.

The car-sequencing problem corresponding to Figure 1 is shown in Figure 2, and like the Hamiltonian Path problem has no solution. For example, the sequence $n_1, e_1, v, n_2, e_2, v, n_3, e_3, v, n_4, e_4, v, n_5, e_5, v, n_6, e_6$ fails just one capacity constraint, because cars e_4 and n_5 both require option E_5 which may not occur twice in three cars. If we add an edge between nodes 4 and 5 in the example graph, e_4 no longer requires E_5 and the reader can check that the same sequence now satisfies all the capacity constraints.

2.3 Proof

The translation above will allow cars to be sequenced in either order, and I believe any car-sequencing solution can be reversed and remain a valid solution. So without loss of generality (WLOG) I will assume that the order of the cars is from node to destination. Note that the translation is easily polynomial in the size of the HP problem.

	configs	n_1	e_1	n_2	e_2	n_3	e_3	n_4	e_4	n_5	e_5	n_6	e_6	v
number	of cars	1	1	1	1	1	1	1	1	1	1	1	1	5
Option	p/q													
C_1	1/16	x											x	
C_2	1/14		x									x		
C_3	1/3	x		x		x		x		x		x		
C_4	1/3		x		x		x		x		x		x	
T_1	1/2	x			x		x		x		x		x	
T_2	1/2		x	x			x		x		x		x	
T_3	1/2		x		x	x			x		x		x	
T_4	1/2		x		x		x	x			x		x	
T_5	1/2		x		x		x		x	x			x	
T_6	1/2		x		x		x		x		x	x		
E_1	1/3	x					x		x					
E_2	1/3			x									x	
E_3	1/3		x			x					x		x	
E_4	1/3		x					x			x		x	
E_5	1/3						x		x	x				
E_6	1/3				x		x		x			x		

Figure 2: The unsolvable car sequencing problem derived from the example graph. Addition of the edge 4-5 changes corresponds to removal of option E_5 from e_4 and of option E_4 from e_5. As a result the car-sequencing problem can be solved with all cars n_i and e_i in the order shown, with the five cars v being placed between each e_i and n_{i+1}.

The following lemmas relate to any solution to the car sequencing problem translated from an HP problem.

Lemma 1 *If the capacity constraints on options C_1 and C_2 are satisfied, the first car sequenced is the node-car for the source, the second the edges-car for the source, the penultimate car is the car-node for the destination and the last car the edges-car for the destination.*

Proof. Option C_1 allows only one car in $3N-2$, so the two cars needing this option must be at least $3N-2$ cars apart in the sequence, and we only have $3N-1$ cars in total. So the only way to allow this is to put these two cars at either end of the sequence of cars. WLOG assume that the node-car for the source is first. Then the edges-car for the destination must be last. This only leaves $3N-3$ other cars, and so we can use a similar argument to show that the node-car for the destination and the edges-car for the source must be at either end of the remaining cars. Finally note that we cannot start the sequence with two node-cars because of option C_3.

Lemma 2 *Nodes car appear exactly in positions $1, 4, 7, \ldots 3N-2$. Edges-cars appear exactly in positions*
$2, 5, 8, \ldots 3N-1$.

Proof. Lemma 1 tells us this is correct for positions 1, 2, $3N-2$ and $3N-1$. Option C_3 allows only 1 in 3 cars, so the next cannot be before position 4 and the last not after $3N-5$. There are $N-2$ node-cars remaining which need option C_3 to put into this subsequence of length $3N-8$. In general, the minimum length of sequence necessary to put n cars with capacity constraint $1/m$ is $(n-1)m+1$. In the case of $n = N-2$ and $m = 3$ the minimum length is $(N-3)3+1 = 3N-8$, exactly the length of sequence we have. The only way to do this is to put the cars in every third position starting from 4. A similar argument puts the edges-cars into the places stated.

Lemma 3 *The edges-car e_i immediately follows the node-car n_i.*

Proof. From Lemma 2 every node-car is followed by some edges-car. The car n_i requires option T_i as does every e_j *except* e_i. Since option T_i may not be given to consecutive cars, e_i is the only car that may follow the n_i.

Lemma 4 *If node-car n_j appears 3 cars after car n_i there is an edge from i to j in the original graph.*

Proof. From Lemma 3 the node-car for i is followed by the edges-car for i. So the node-car for j appears 2 cars later than this edges-car. But the node-car for j needs option E_j, and this may only appear in 1 of every 3 cars. So the edges car for i cannot need option E_j, which by definition means that there is an edge from i to j in the original graph.

Theorem 5 *If the translated car-sequencing problem has a solution, then so does the original Hamiltonian Path problem.*

Proof. By definition of the car-sequencing problems, all the node-cars are sequenced in a solution. Consider the sequence of nodes in the original graph defined by the order of corresponding node-cars in the car-sequencing solution. By Lemma 1 the source node is first and the destination node last. By Lemma 2 the node-cars appear every three cars after the source node-car. And by Lemma 4, there is an edge in the original graph between each two nodes corresponding to consecutive cars in this subsequence of node-cars. To summarise, a car-sequencing solution defines a permutation of nodes in the original graph starting from the source, with an edge between each consecutive pair of nodes, and ending at the destination. Which was what was wanted.

We must prove the converse of this theorem to finish the proof of NP-completeness. The following theorem does this.

Theorem 6 *If the original HP problem has a solution, so does the translated car-sequencing problem.*

Proof. We have some permutation π of the nodes in the original graph forming a successful path from source to destination. Suppose that node i appears at location $\pi(i)$ in this permutation, so that for example $\pi(\text{source}) = 1$ and $\pi(\text{destination}) = N$. Then I claim that we solve the the car-sequencing problem if we put the node-car for i in position $3\pi(i) - 2$, the edges-car for i in position $3\pi(i) - 1$, and vanilla cars in all remaining positions. We now need to check that all the capacity constraints will be satisfied.

First look at the control options. By construction the two cars needing C_1 occur at positions 1 and $3N - 1$. These satisfy the capacity constraint of $1/3N - 2$. Similarly the two cars needing C_2 satisfy the capacity constraint of $1/3N - 4$ by occurring at positions 2 and $3N - 2$. The 1/3 constraint C_3 is satisfied because the only cars needing it are node-cars which occur at intervals of 3, and the same comment for edges-cars applies to contraint C_4.

Next look at the options tying nodes and edges, of capacity 1/2. Because edges-cars are always 3 apart and vanilla cars need no options, we need only worry about clashes between consecutive node and edges-cars. But car n_i only takes option T_i of the tying options, and n_i is followed by car e_i which by construction does not require T_i. Therefore the capacity constraint on every option T_i is satisfied.

Finally we must check the options E, all of capacity 1/3. Each node-car n_i only takes option E_i, and is immediately followed by e_i and a vanilla car, neither of which needs E_i. So the only way that a constraint E_i can fail is if the edges car e_i and the node-car n_j two cars later both need E_j. But by construction we must have that $\pi(j) = \pi(i) + 1$ and since π represents a Hamiltonian path, there is an edge in the graph between nodes i and j. So by the definition of the translation, the edges car e_i does not require option E_j.

3 Solution to an Open Car Sequencing Problem

Régin and Puget [7] introduce a filtering technique and describe its use on hard car-sequencing problems proposed by Smith [8]. They solve four, of which two are solvable and two not. One problem remained open. To their credit, Régin and Puget list the problem in full and mention their failure, even emphasising it in their talk at CP-97. Such reporting of negative results is to be encouraged.

The problem is as described in the following table. The top row just lists the identifiers for the configurations, while the bottom row gives the number of cars required in each configuration. There are 5 options, 23 configurations using those options, and 100 cars to be built in total. The five central rows in the table detail the name of the option, the capacity constraint, and the configurations which require that option.

	configs	**0**	1	2	**3**	4	5	**6**	7	8	9	10	11	**12**	13	14	**15**	16	17	18	**19**	20	21	22
i	p_i/q_i	$\downarrow$			$\downarrow$			$\Downarrow$						$\Downarrow$			$\Downarrow$				$\downarrow$			
0	1/2						x	x			x	x		x	x	x	x	x	x				x	
1	2/3			x		x	x	x		x	x			x		x	x			x		x	x	x
2	1/3		x	x				x	x				x	x	x		x			x				x
3	2/5	x		x	x	x					x		x	x	x			x					x	
4	1/5	x	x				x	x											x		x	x	x	x
	cars	2	2	5	4	4	1	3	4	19	7	10	1	5	2	6	4	8	1	4	2	4	1	1

3.1 Proof of Insolubility

The proof that the problem is insolvable depends only on the six columns marked by boldface and the arrows, i.e. configurations 0, 3, 6, 12, 15 and 19. The key point is that the columns marked $\Downarrow$, i.e. 6, 12 and 15 , require all of options 0, 1, and 2. The columns marked $\downarrow$ are the only configurations which involve none of 0, 1, and 2.

What can be adjacent to a $\Downarrow$-car? Option 0 is 1/2 so a $\Downarrow$-car cannot be preceded or followed by another option 0-car. Similarly option 2 is 1/3 so no $\Downarrow$-car can be adjacent to any option 2-car, and in fact we cannot even put two option 2-cars with only one other car between them. Finally, option 1 is 2/3, so unless a $\Downarrow$ is either the first or the last car, it must be adjacent to a car which does not need option 1. This car not needing option 1 cannot have 0 or 2 either, so can only be given options 3 or 4, i.e. be a $\downarrow$-car.

There are 12 cars in total marked by $\Downarrow$. One can be put first and one last in the sequence, leaving 10. For each of these 10 we need at least one $\downarrow$-car to be adjacent: we cannot sandwich a $\downarrow$-car between two $\Downarrow$-cars because of option 2.

Unfortunately, there are only 8 $\downarrow$-cars, and we need 10. So the problem is insoluble.

4 Conclusions

The solution of an open benchmark does not have dramatic consequences for the field although it does clarify the status of the benchmark set. In particular, it confirms that previous attempt to find a solution failed for the simple reason that no solution existed. It would have been rather more disconcerting if hand methods had found a solution to this problem, as has happened for example on bin-packing benchmarks [3]. Computer methods often fail on large combinatorial problems such as this one, although it is certainly interesting that this problem has a small proof of impossibility.

I have proved that the Car Sequencing Problem is NP-complete. This is hardly surprising, but the language just might have been restricted enough to make it polynomial. As Christos Papadimitriou wrote, ‘An NP-completeness proof is typically the first act of the analysis of a computational problem by the methods of the theory of algorithms and complexity, not the last’

[4]. NP-complete problems are not all the same: problems can be strongly NP-complete, or not, and have a variety of approximation properties. Pursuing such questions leaves many avenues open for further work.

Acknowledgements

I am a member of the cross-site APES Research Group[1]. I thank fellow members from both Leeds and Strathclyde Universities, particularly Ewan MacIntyre, Barbara Smith, and Toby Walsh. I also thank Jean-Francois Puget from ILOG.

References

[1] M. Dincbas, H. Simonis, and P. Van Hentenryck. Solving the car-sequencing problem in constraint logic programming. In *ECAI-88*, pages 290–295, 1988.

[2] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. W H Freeman, 1979.

[3] I.P. Gent. Heuristic solution of open bin packing problems. *Journal of Heuristics*, 3(4):299–304, 1998.

[4] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[5] B.D. Parello. CAR WARS: the (almost) birth of an expert system. *AI Expert*, 3(1):60–64, 1988. Cited in [1].

[6] B.D. Parello, W.C. Kabat, and L. Wos. Job-shop scheduling using automated reasoning. *Journal of Automated Reasoning*, 2(1):1–42, 1986. Cited in [1].

[7] J.-C. Régin and J.F. Puget. A filtering algorithm for global sequencing constraints. In *Principles and Practice of Constraint Programming - CP-97*, pages 32–46. Springer, 1997. LNCS 1330.

[8] B.M. Smith. Succeed-first or fail-first: A case study in variable and value ordering. In *Proceedings of PACT'97*, pages 321–330, 1997.

[1] `http://www.cs.strath.ac.uk/~apes`