Optimizing the Standardization of Car Cables

Marie-Sklaerder Vié[1], Nicolas Zufferey[1,2], Jean-François Cordeau[2,3]

[1] Geneva School of Economics and Management, GSEM - University of Geneva, Switzerland
[2] Groupe d'Études et de Recherche en Analyse des Décisions, GERAD, Canada
[3] École des Hautes Études Commerciales, HEC Montreal, Canada
marie-sklaerder.vie@unige.ch, n.zufferey@unige.ch, jean-francois.cordeau@hec.ca

Key-words : *Combinatorial optimization, Production, Heuristics*

Abstract : *A set-partitioning problem for car wirings is addressed, in order to minimize the production costs. An approach to divide the problem in smaller sub-problems is presented, and three different constructive methods are proposed and compared according to quality, computing time, and robustness.*

1 Introduction

Due to the wide variety of vehicles, car manufacturers have a huge number of different electrical services to build. Indeed, they could make as many wirings (where a wiring is made of a set of modules, each module having a different presence cost if it appears in the wiring) as there are electrically different cars, but this brings diversity management costs, called the *diversity tax*. Alternatively, they could make a few standardized wirings, rich enough to suit different vehicles, even if unnecessary modules are set up on many cars, which has also an important cost, called the *give-away*. In any case, some constraints have to be satisfied. On the one hand, some modules require the presence of other modules (*implication* constraints), and on the other hand, some modules are incompatible (*exclusion* constraints). The goal of this work is to find a balance between these two extreme solutions, hence minimizing the costs (composed of the give-away plus the diversity tax), while covering the demand, and satisfying the implication/exclusion constraints.

The considered problem, denoted as (P), is NP-hard. Indeed, it is a generalization of problems as the *Generalized Assignment Problem* [6] and the *p-Median Problem* [5], which are known to be NP-hard [3]. Real data will be considered, obtained from the French multinational car manufacturer Renault (in 2013 it was the eleventh biggest automaker in the world by production volume, with 50.5% of sales coming outside of Europe). We consider instances covering 67% of 7 years of production for Renault, including past, present and future years (one set covering 6 months for just a section of a car i.e., either the dashboard, the engine, the ceiling, the trunk, or the doors, from the most to the less complex section). On the one hand, the considered data sets are too big for CPLEX. On the other hand, a simpler problem (where the produced wirings are not created but chosen among the demanded ones) was investigated in [1] with an exact method, but it was limited to small instances (up to roughly 5000 demands). Therefore, heuristics and metaheuristics appear as the most appropriate methods. The reader is referred to [4] for a good survey on metaheuristics, and to [7] for guidelines on how to design them efficiently.

The contributions of this paper are: (1) a new but real problem (P) is formulated, and a way to divide it into sub-problems with fewer constraints (Section 2); (2) constructive solution methods are proposed (Section 3). The results are presented in Section 4 and a conclusion ends the paper in Section 5.

2 Formulation and decomposition of (P)

A set of demands $\mathcal{D}$ is given, of typical size in $[1\,000, 50\,000]$. Each demand $d \in \mathcal{D}$ is made on the one hand of a list $\mathcal{M}_d$ of the different requested modules among the set $\mathcal{M}$ of all possible modules (of typical size in $[20, 60]$), corresponding to the electrical options that the client (which is not a final consumer but a car dealer) wants for its vehicles, and on the other hand of the number Q_d of vehicles it wants, with typical values in $[1, 30\,000]$. Producing each demand separately would be too expensive

for Renault, because of the huge resulting diversity tax. An *envelope* (wiring) is a given set of modules, that is usually large enough to cover several demands. For each given instance, each demand is covered by exactly one envelope (i.e., all the modules requested in a demand belong to its assigned envelope), in order for Renault to know that, even if those envelopes are expensive (regarding the give-away), a feasible solution is to build them only. The envelopes express technical barriers (e.g., right or left steering), which means that two different envelopes cannot be grouped together as one wiring. In other words, the envelopes represent implicit exclusion constraints. The set of envelopes is denoted $\mathcal{E}$, and has a typical size in $[20, 40]$. A *reference* (wiring) is defined the same way as an envelope, but it has to be built, each reference built being covered by exactly one envelope. The goal of building these references is to have another set of wirings, that also covers all the demands, but that has a smaller production cost than the envelope set (indeed, the give-away will be reduced, but the diversity tax will be augmented). The references have to respect some (less than 10 for every instance) implication and exclusion constraints for their modules (i.e., if some modules are present in a reference, others must be too, or others cannot be). Note that the give-away can be a non-linear function, because some modules can have different prices if they are together or not in a reference, and some groups of modules have a fixed price if one or more modules of this group are in a reference. The resulting combinatorial optimization problem (P) consists in building a reference set $\mathcal{S}$ that minimizes the production costs, defined as the give-away plus the diversity tax.

As the give-away is growing with the size $|\mathcal{S}|$ of $\mathcal{S}$, and the diversity tax is decreasing with it, the production cost is a convex function. Therefore, if this function is known, its minimum would be easy to find. As the diversity tax is part of the given data (i.e., for each allowed value N of $|\mathcal{S}|$, its associated diversity tax is known), the remaining problem is to build the give-away function. More precisely, the give-away has to be minimized for each possible value of $|\mathcal{S}|$, where Renault imposes to have $|\mathcal{S}| \leq 4 \cdot |\mathcal{E}|$. This kind of decomposition (i.e., optimizing each level independently) has been proven to be efficient for many optimization problems. In addition, this decomposition allows Renault to have a solution for each possible value N of $|\mathcal{S}|$, which gives them more flexibility in their production afterwards, for example if the demands change (as some instances are based on forecast demand).

Using the fact that each demand is covered by only one envelope, and that envelopes represent exclusion constraints (i.e., the modules of any two demands covered by different envelopes are incompatible), (P) can be decomposed into different sub-problems, one for each envelope. This can be done by grouping the demands covered by the same envelope. Formally, for each demand $d \in \mathcal{D}$, one and only one envelope $e \in \mathcal{E}$ covers d. It means that a reference r cannot cover two demands that are not covered by the same envelope. For every envelope $e \in \mathcal{E}$, the set of demands covered by e is denoted by $\mathcal{D}_e = \{d \in \mathcal{D} \mid e \text{ covers } d\}$. As a consequence, (P) can be decomposed into sub-problems of type (P_e) restricted to $\mathcal{D}_e$ instead of $\mathcal{D}$. The different sub–problems will only be linked by the constraint on the total number N of references to use of the whole instance. If a solution $\mathcal{S}_e$ is found for each (P_e) using a number N_e of references, a solution $\mathcal{S}$ of (P) is therefore $\mathcal{S} = \bigcup_e \mathcal{S}_e$, with $\sum_e N_e = N$.

3 Solution methods for (P)

Two different greedy algorithms are proposed. First, the *Demands Greedy Algorithm* ($Gr^{\mathcal{D}}$) is a bottom-up approach that starts from the demands as initial references, and groups them two by two following the cheapest grouping cost until the number N of references is reached. The grouping cost $f^g(r, r')$ is the increase of the give-away if two references r and r' are grouped together. Second, the *Envelopes Greedy Algorithm* ($Gr^{\mathcal{E}}$) is a top-down approach that starts from the envelopes and creates the references by separating the corresponding demands in two parts, following the best separation cost (ties are broken randomly) until the number N of references is reached. The separation cost $f^s(r, m)$ is the decrease of the give-away if a reference r containing a module m is split into two references r_m and $r_{\bar{m}}$ such that: r_m covers each demand (covered by r) that contains m, and $r_{\bar{m}}$ covers each demand (covered by r) that does not contain m. Both algorithms follow the same pattern, described in Algorithm 1. Each step is described for $Gr^{\mathcal{D}}$, and then for $Gr^{\mathcal{E}}$ (separated with "/") if the algorithms differ.

In order to explore more solutions, using a less deterministic constructive algorithm, a *Greedy Randomized Procedure* (GRP) was developed, inspired from the *Greedy Randomized Adaptive Search Procedure* (GRASP) [2]. This algorithm follows exactly the same steps as a greedy algorithm, but instead of making

at each iteration the best possible decision, it chooses randomly one among the p best ones (parameter tuned to 3 after preliminary experiments). This approach was tested within $Gr^{\mathcal{E}}$ only, because many ties are already encountered in $Gr^{\mathcal{D}}$ (remind that ties are always broken randomly).

Algorithm 1 $Gr^{\mathcal{D}}$ (bottom-up) / $Gr^{\mathcal{E}}$ (top-down)

Initialization (for each $e \in \mathcal{E}$**)**

1. For $Gr^{\mathcal{D}}$, set $\mathcal{S}_e = \mathcal{D}_e$ / For $Gr^{\mathcal{E}}$, set $\mathcal{S}_e = \{e\}$.
2. Compute all the grouping / separation costs.
3. Set f_e as the: grouping cost of the pair (r, r') of references minimizing f^g (give-away augmentation) / separation cost of the reference r and module m maximizing f^s (give-away reduction).

While the number N of references is not reached, **do**:

1. For the envelope e with the optimal cost f_e, group / separate the corresponding references.
2. If the total number $\sum_e N_e$ of references in all the sub-problems has reached N, then stop.
3. Otherwise, update the grouping / separation costs affected by the last decision.

4 Results

Table 1 presents the results. For each instance, the characteristics (size, const, costs) are first indicated. The size is said small if the larger sub-problem has less than 1 000 demands, medium if it has less than 10 000 demands, and large otherwise. The column "const" only states if the instance has implication/exclusion constraints or not. The column "costs" states if there are non-linear production costs or not (as explained in Section 2). Next, for each algorithm, three columns describe its results averaged over ten runs. The first one gives the average percentage gap (for each possible number N of references) with respect to the best solution value found by the best run among all the algorithms, the second one gives the average computing time, and the third one gives the standard deviation σ (again, averaged over the possible numbers of references).

The results show that $Gr^{\mathcal{D}}$ appears to almost always find the best solutions, but $Gr^{\mathcal{E}}$ finds solutions that are not too far from the best ones, and in a computing time that can be up to fifty times faster. Both $Gr^{\mathcal{D}}$ and $Gr^{\mathcal{E}}$ are robust (the average standard deviation being usually zero). Finally, GRP manages sometimes to improve the solution of $Gr^{\mathcal{E}}$, but on average only for two instances, and is less robust than the two other algorithms (even if the deviation is not that important, it is enough to impose more than one run in order to find the best possible solution with this algorithm).

A typical graph obtained by the three algorithms for one instance is given in Figure 1. For each allowed number N of references, it shows the best value of the average give-away (averaged over all demands and over 10 runs). The same conclusions can be made with this graph: $Gr^{\mathcal{D}}$ gives the best solutions, and $Gr^{\mathcal{E}}$ and GRP give similar solutions, not too far from the ones provided by $Gr^{\mathcal{D}}$.

5 Conclusion and future works

Three different constructive methods were developed to tackle the combinatorial optimization problem (P) faced by Renault. Its two main difficulties are the huge size of the real instances and the non-linearity of the costs and constraints, which together make impossible to solve (P) exactly. Nevertheless, the three proposed algorithms offer solutions in a reasonable computing time. $Gr^{\mathcal{D}}$ offers the best solutions for almost all the instances, hence it is recommended if there is no computing-time constraint. Otherwise, $Gr^{\mathcal{E}}$ offers solutions with almost the same cost, but much quicker. Among the next steps, we propose to use $Gr^{\mathcal{E}}$ to initialize a local search or other metaheuristics.

TAB. 1 – Performance of the Greedy Algorithms

Instances			$Gr^{\mathcal{D}}$			$Gr^{\mathcal{E}}$			GRP		
size	**const**	**costs**	**%gap**	**time**	σ	**%gap**	**time**	σ	**%gap**	**time**	σ
small	no	lin	0.00	0	0.00	46.57	0	0.00	43.12	1	0.16
small	yes	lin	0.00	1	0.00	41.07	0	0.00	56.93	0	0.79
small	yes	lin	0.00	4	0.00	10.10	1	0.00	8.71	1	0.47
small	yes	non-lin	0.00	1	0.00	1.50	1	0.00	7.19	1	0.23
small	yes	non-lin	0.00	2	0.00	69.41	1	0.00	81.97	1	0.31
small	yes	non-lin	0.00	4	0.00	69.45	1	0.00	72.89	1	0.45
small	yes	non-lin	0.00	10	0.00	15.15	2	0.00	17.34	2	0.18
small	yes	non-lin	0.00	0	0.00	362.47	0	0.00	533.29	0	0.18
medium	no	lin	0.00	8	0.01	6.61	1	0.00	13.02	1	0.18
medium	yes	lin	0.00	12	0.00	8.79	1	0.00	18.17	3	0.17
medium	yes	lin	0.00	30	0.04	42.55	2	0.00	30.84	2	0.15
medium	yes	lin	0.00	60	0.04	9.53	5	0.00	8.95	5	0.15
medium	yes	lin	0.00	77	0.00	8.79	28	0.00	9.24	28	0.07
medium	yes	lin	0.00	130	0.02	38.79	10	0.00	34.44	10	0.11
medium	yes	lin	0.00	149	0.02	16.82	12	0.00	21.54	13	0.14
medium	yes	non-lin	0.00	9	0.00	48.24	1	0.00	49.17	1	0.21
large	no	lin	1.16	1437	0.23	1.86	29	0.00	0.23	31	0.43
large	no	lin	0.00	2078	0.18	10.63	44	0.00	6.58	46	0.47
large	yes	lin	0.00	1738	0.06	16.57	35	0.00	15.18	35	0.18
large	yes	non-lin	0.00	2599	0.05	9.11	49	0.00	9.67	48	0.12
Average			**0.06**	**418**	**0.03**	**41.70**	**11**	**0.00**	**51.92**	**11**	**0.26**

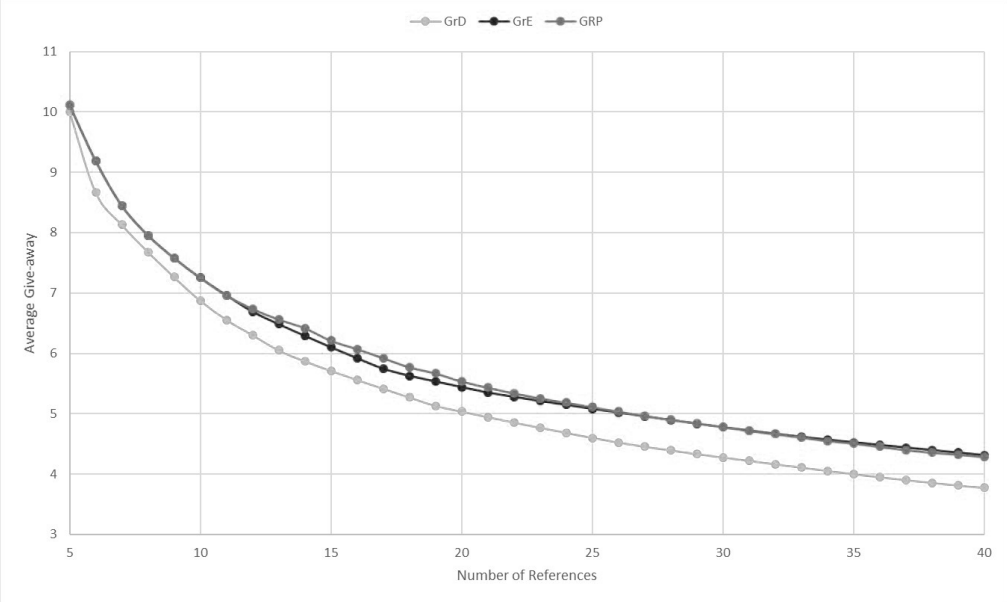

FIG. 1 – Average give-away obtained for one instance

References

[1] O. Briant and D. Naddef. The Optimal Diversity Management Problem. *Operations Research*, 52(4):515–526, 2004.

[2] T.A. Feo and M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[3] M.R. Garey and D.S. Johnson. *Computer and intractability.* W. h. free edition, 1979.

[4] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics.* Springer edition, 2010.

[5] N. Mladenović, J. Brimberg, P. Hansen, and J.A. Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939, 2007.

[6] K. Sethanan and R. Pitakaso. Improved differential evolution algorithms for solving generalized assignment problem. *Expert Systems with Applications*, 45:450–459, 2016.

[7] N. Zufferey. Metaheuristics : Some Principles for an Efficient Design. *Computer Technology & Application*, 3:446–462, 2012.