




Article

An Ant Colony Algorithm for Improving Energy Efficiency of Road Vehicles

Alberto V. Donati ^{1,*}, Jette Krause ¹, Christian Thiel ¹ , Ben White ²  and Nikolas Hill ³ 

¹ Joint Research Centre of the European Commission, Via Fermi 2749, 21027 Ispra (VA), Italy; jettekrause@aol.com (J.K.); christian.thiel@ec.europa.eu (C.T.)

² Ricardo Energy and Environment, 30 Eastbourne Terrace, Paddington, London W2 6LA, UK; b.white.1@hotmail.co.uk

³ Ricardo Energy and Environment, Gemini Building, Fermi Avenue, Harwell, Oxon OX11 0QR, UK; nikolas.hill@ricardo.com

* Correspondence: alberto.donati@ext.jrc.ec.europa.eu

Received: 14 May 2020; Accepted: 30 May 2020; Published: 3 June 2020



Abstract: The number and interdependency of vehicle CO₂ reduction technologies, which can be employed to reduce greenhouse emissions for regulatory compliance in the European Union and other countries, has increasingly grown in the recent years. This paper proposes a method to optimally combine these technologies on cars or other road vehicles to improve their energy efficiency. The methodological difficulty is in the fact that these technologies have incompatibilities between them. Moreover, two conflicting objective functions are considered and have to be optimized to obtain Pareto optimal solutions: the CO₂ reduction versus costs. For this NP-complete combinatorial problem, a method based on a metaheuristic with Ant Colony Optimization (ACO) combined with a Local Search (LS) algorithm is proposed and generalized as the Technology Packaging Problem (TPP). It consists in finding, from a given set of technologies (each with a specific cost and CO₂ reduction potential), among all their possible combinations, the Pareto front composed by those configurations having the minimal total costs and maximum total CO₂ reduction. We compare the performance of the proposed method with a Genetic Algorithm (GA) showing the improvements achieved. Thanks to the increased computational efficiency, this technique has been deployed to solve thousands of optimization instances generated by the availability of these technologies by year, type of powertrain, segment, drive cycle, cost type and scenario (i.e., more or less optimistic technology cost for projected data) and inclusion of off-cycle technologies. The total combinations of all these parameters give rise to thousands of distinct instances to be solved and optimized. Computational tests are also presented to show the effectiveness of this new approach. The outputs have been used as basis to assess the costs of complying with different levels of new vehicle CO₂ standards, from the perspective of different manufacturer types as well as vehicle users in Europe.

Keywords: CO₂ reduction; multi-objective combinatorial optimization; meta-heuristics; ant colony optimization

1. Introduction

In recent years, the automotive industry has faced increasing pressure to reduce CO₂ emissions to meet regulatory targets set in both the EU and other legislations. Several new CO₂ reduction and energy efficiency improvement technologies have been developed to respond to these new regulatory developments. This technology proliferation increases the complexity of finding optimal combinations to achieve substantial CO₂ reductions in a cost-efficient way. The problem consists of finding feasible configurations, represented by points in the plane total CO₂ reduction versus price,

and identifying the limiting subset of those, which form the Pareto front, to yield the optimal configurations for the objectives defined. The optimization problem is computationally difficult because of the size of the search space. Because of incompatibilities between certain technologies, the optimization problem cannot be solved by simple sorting according to technology efficiency (CO₂ reduction per cost) and constructing configurations starting by the most efficient technology, adding technologies based on their efficiency, from the most efficient to the least.

In the following, we discuss similar problems found in the literature and various approaches to tackle them. The Knapsack Problem (KP) and its multi-dimensional variant (MKP) are problems aimed to find an optimal subset of objects to be placed in one or more knapsacks (or limited capacity boxes/containers), maximizing the total profit or utility. These problems belong to the class of NP-complete problems and have been extensively explored as a fundamental area of discrete optimization and Operations Research (OR). Whilst the standard KP seeks to maximize the utility under a single constraint, other variations or generalizations can deal with the assignment of resources, such as in freight logistics planning, capital budgeting, allocation of tasks on processors, stock cutting problems or in distributed computing as presented in [1,2]. A well-studied extension of the MKP is the Multi-Objective Multi-dimensional Knapsack Problem (MOMKP), when taken with two objective functions and no constraints on the number or capacities of the knapsacks. A detailed survey of techniques to solve the Multi-Objective Knapsack Problem (MOKP) and MOMKP is presented in [3]; it discusses exact methods, approximation algorithms and heuristics for the MOKP, including Simulated Annealing, Tabu Search and other heuristic methods, subdivided into evolutionary algorithms and those based on local search (LS). Other methods, such as Fuzzy Logic and Kalman Filtering have been successfully applied in detecting failures of integrated sensors/systems on vehicles where high-dimensional data is present [4]. State-of-the-art genetic algorithms have been extensively applied to the multi-objective optimization like in [5,6], to the multi-criterion optimal design [7].

The Technology Packaging Problem (TPP) is introduced here since the problem differs considerably from a MOMKP, in that, if a knapsack is assimilated to a package or configuration:

- (1) The same object can be placed in one or more knapsacks (or configurations), but with no repetition within the package,
- (2) some objects cannot be combined in the same knapsack (or package) due to the incompatibility between them, so one or the other shall be chosen,
- (3) there are no constraints on the number and capacity of knapsacks, in the sense that the number of the feasible optimal packages/configurations is not set a priori, nor its costs (or capacity) is given a priori, but rather found by the optimization.

The difference described in point (3) can be particularly noticeable if one would try to approach the problem by fixing one objective and trying to optimize the other. More specifically, this could be done by partitioning the cost (the knapsack capacity) from a minimum to a maximum cost, say in 100 slices, and solve the same number of optimization problems each with the maximum cost of the slice considered. However this would result in a loss of solutions, since the optimization would end with a pareto front with exactly the same number of solutions as slices and, further, a potential loss of efficiency in having to re-compute from scratch the solution for each slice. This would exclude for example the possibility of incrementally adding objects/technologies to already found optimal configurations. The reduction of the 2-objective problem to a sliced single objective problem has some drawbacks which will be discussed in Section 4. On the other hand, we have attempted the reduction of the current problem to other similar multi objective and constrained problems, but none brought to a satisfactory formulation without a significant loss of details or changes in the original problem. Hence, the introduction of the Technology Packaging Problem (TPP). To solve this problem with two (or more) conflicting objectives, we have devised an Ant Colony Optimization (ACO) combined with a Local Search heuristic (ACO + LS).

Bio-inspired algorithms have gained consensus as being some of the most efficient and effective to deal with complex problems and constraints in various areas of Operational Research. Examples include evolutionary algorithms, swarm intelligence, distributed systems, neural networks and similar heuristic/meta-heuristics methods for NP-complete or other computationally hard problems. In particular, the choice of ACO is motivated by the recent success of swarm intelligence approaches to solve complex and dynamical problems with a system of interacting agents, which combines exploration and learning (ACO) with simple local rules (LS). These approaches give rise to a collective and distributed intelligence through a form of system memory whereby global information is encoded locally, as presented extensively in [8,9]. ACO heuristics have proven to be particularly effective when combined with appropriate local search techniques, as evidenced in [10] and [11]. They are commonly used in operational research problems such as the classic Traveling Salesman Problem (TSP), quadratic assignment and job-shop scheduling. ACO extends to applications in problems of a dynamic nature due to the algorithm's adaptability and re-scheduling capabilities [12], for example the Vehicle Routing Problem (VRP) with variable traffic conditions [13] or with unexpected events [14]. In [15] ACO is used for the Multi-Stage Flow Shop Scheduling Problem and applied for the scheduling of real factories. The introduction of multiple colonies and different pheromones update strategies has been proved effective for multi-objective problems. An example is the application to the time and space assembly line balancing problem, presented in [16], where eight different ACO architectures are introduced and compared.

One of the first applications of ant colony inspired algorithms to the MKP and the Subset Problem (SP)—a special case of the knapsack problem—was made by [17] with an Ant System, a variation of ACO incorporating two distinct pheromone updating rules: local and global. This was followed specifically for the MKP by [18,19] (where a general ACO is introduced for multi-objective optimization) and later by [20], reporting very successful results in comparison to other evolutionary algorithms. The MKP, SP and TPP are quite different from TSP-like problems, as the ordering of objects or tasks to complete is not important: for the former the solutions are combinations, yet for the latter they are permutations (ordered sequences).

Some further considerations are therefore needed in the representation of the problem and in particular regarding the pheromone encoding. Two main approaches are present in the literature: node-based, deposited on the node or edge-based, deposited on the edges connecting nodes/objects. In this study we adopted the former, as presented in [18].

The TPP is formulated in Section 2, and the ACO + LS algorithm is discussed in Section 3. A short description of the construction of so-called cost curves for vehicle CO₂ emission reduction is also given. In Section 4, the ACO + LS results are compared with those obtained previously with a Genetic Algorithm (GA). The real world optimization problem instances, discussed in Section 5, consist of a set of specific technologies which are available and a set of initialization parameters; these include the type of vehicle, its size and powertrain, the year considered and the drive cycle over which the technology is evaluated, which together act to determine the presence or absence of a technology. Finally conclusions are drawn in Section 6.

The contribution of this study is in the following:

- (a) The formulation of a novel OR problem, the TPP, which has not been seen in existing literature, which has with some crucial differences from similar problems as discussed at points (1) to (3) above this section.
- (b) Since the problem is new, the development of a method for performing the optimization once the TPP is applied to the two objective optimization problem of vehicular CO₂ reduction technologies.
- (c) The evaluation of the computational efficiency of this method on an extended set of problem instances.
- (d) The extensive application of the method to thousands of different instances, each representing a vehicle type (size, segment, power train, etc.) to provide, for each vehicle type, its “cost curve” to

be used for each manufacturer type's fleet composition and the combined CO₂ emissions of its circulating vehicles.

With the above, once ran over the entire EU vehicle fleet typology has been used as an input for future CO₂ targets, so one of its most relevant contributions has been to the support of the EU policy making to check the feasibility of the 2025–2030 CO₂ reduction targets. The cases presented in Section 5 are just a part of a number of simulations/optimizations ran by the authors. Most of these, and their rationale, have been omitted since they are beyond the scope of this study. Thanks to its efficiency, the system has been used extensively for scenario analysis, for example to find configurations with certain technologies which are always present (such as where these technologies have already been embedded in the vehicles) or for the analysis in which different future scenarios for the price projection or CO₂ reduction potentials were to be used. Overall, the authors have run about 60,000 problem instances in the course of these analyses.

2. The Optimization Problem

The TPP with two-objectives is defined as follows. Given:

- (1) A set of N technologies $T = \{t_1, \dots, t_N\}$, each characterized by two numerical values (f_i, g_i) of their utility (or inverse utility, like cost, in the case of minimization), for every $i = 1, \dots, N$.
- (2) A matrix $E = \{e_{ij}\}$, $e_{ij} = \{0, 1\}$, defining the incompatibilities between technologies t_i, t_j of the above set (with the convention: 1=incompatible, 0=compatible). E is symmetric due to the fact that "incompatibility" is symmetric: if t_i is incompatible with t_j , t_j is incompatible with t_i .
- (3) A package rule: objects can be aggregated in so-called packages or configurations, by combining any subset of compatible technologies $T_k = \{t_{k1}, \dots, t_{kn}\}$ with $k_h \in [1, N]$, containing each technology at most once (no repetition) in each package; subscript indexes k_h are introduced just to indicate that any subset of T is possible, provided the technologies are compatible.
- (4) Two analytical functions, the objectives, to aggregate the utilities of the technologies in the package $T_k = \{t_{kh}\}$: $F = F(T_k) = F(f_{k1}, \dots, f_{kn})$ and $G = G(T_k) = G(g_{k1}, \dots, g_{kn})$, with $k_h \in [1, N]$, to provide the aggregate measure of the package. The functions F and G can assume any problem-specific form, to appropriately combine the utilities of the technologies/objects composing a given configuration. They can simply be the sum of utilities or their product.

The problem is then to find, among all the possible packages/configurations, those which are Pareto optimal with respect to two objective functions chosen. Since the technologies considered here are CO₂ reduction technologies, each with a cost c_i and CO₂ reduction potential r_i (which represents the CO₂ emission reduction percentage value that can be attained with the technology in place), the objective functions F and G can be defined as the total cost of the package, C_k and its total CO₂ reduction R_k as follow:

$$C_k = \sum_{t_i \in T_k} c_i \quad (1)$$

$$R_k = 1 - \prod_{t_i \in T_k} (1 - r_i) \quad (2)$$

Equation (2) is obtained by considering the cumulative, interacting benefits of the technologies. For example, if technology 1 has $r_1 = 0.1$ (i.e., 10%) and technology 2 has $r_2 = 0.05$ (i.e., 5%), the total reduction of the combined technologies is obtained by applying technology 2 only after we have applied technology 1. Since technology 1 will provide a CO₂ emission of $(1-r_1)$, applying the second will result in a total CO₂ emission of $(1-r_1) \times (1-r_2)$. The total reduction of applying the two technologies is therefore $R = 1 - (1-r_1) \times (1-r_2)$ (hence the form of Equation (2)), which given r_1 and r_2 is 0.145 or 14.5%.

For this specific application, the optimization is aimed at the minimization of Equation (1) and the maximization of Equation (2). Then the Pareto optimality condition between any two configurations

T_n and T_m (independently of their size) is that: the configuration T_n is said to be dominating the configuration T_m if one of the two following conditions holds:

if:

$$C_n < C_m \text{ and } R_n \geq R_m \text{ or if } C_n = C_m \text{ and } R_n > R_m \quad (3)$$

This means that either T_n has lower total cost and same or higher total CO₂ reduction, or it has the same cost but higher total CO₂ reduction. The configuration T_n is said to be dominated by T_m if one of the following two conditions holds:

if:

$$C_n > C_m \text{ and } R_n \leq R_m \text{ or if } C_n = C_m \text{ and } R_n < R_m \quad (4)$$

If a configuration dominates another one, this cannot belong to the set of the Pareto optimal front. On the other hand, T_n is Pareto optimal with T_m if one of the two following conditions hold:

if:

$$C_n > C_m \text{ and } R_n > R_m \text{ or if } C_n < C_m \text{ and } R_n < R_m \quad (5)$$

Geometrically, these conditions are equivalent to consider in the (R,C) plane, the relative position of the two configurations: $T_m = (R_m, C_m)$ in which to center Cartesian axes, and $T_n = (R_n, C_n)$. If the latter falls in the lower-right quadrant, including the vertical and horizontal semi-axes, then T_n is dominating (condition (3)) T_m , if it falls in the upper-left quadrant, including semi-axes, it is dominated (condition (4)), while in all other cases they are Pareto optimal (condition (5)), so unless other configurations are found to dominate one or both, they are both included in the Pareto optimal set. The set of all Pareto optimal configurations is also referred to as Pareto front.

The size of a configuration is the number of technologies it contains. An exhaustive exploration of solution space, in case of N distinct technologies, would imply the evaluation of (1) and (2) for all possible configurations. The number of all possible configurations of size n , is the combinations of N objects taken n at a time: $\binom{N}{n} = \frac{N!}{n!(N-n)!}$. The total search space size, or the total number of all possible configurations (without considering incompatibilities), is then given by sum of the above expression over all possible configurations' sizes:

$$S = \sum_{n=1}^N \binom{N}{n} = 2^N - 1 \quad (6)$$

where the second equality derives from the Binomial theorem. Even considering incompatibilities (I) between technologies, which tend to diminish the number of the feasible combinations, from a rough estimation of the search space with (6), is where N is usually between 50 and 80 and I is in the order of 2 or 3, is it clear that an exhaustive search is computationally impracticable.

3. The ACO Algorithm

A simple brute force approach would imply testing all possible combinations of the compatible technologies, but this is computationally infeasible, given Equation (6). Some simple heuristics were tested. Technologies were sorted by their cost efficiency (also referred to as 'technology efficiency'), defined as the CO₂ reduction divided by the cost, and to add technologies to the configuration one by one as far as they were compatible with those already present. In the construction process, when an incompatible technology is encountered, it is simply skipped, and the next is examined. This procedure is then repeated, starting each time with a different technology and adding technologies until no further technology could be added.

It was soon noticed that a procedure such as this omits some configurations, since it always starts from a pre-ordering of technologies, based on cost efficiency. Including some randomness greatly improved the number of configurations found. However, a purely random search coupled with a greedy ordering showed to use significant computational power without an efficient search strategy in

the solution space. It was observed that most of the time was spent randomly exploring, rather than reinforcing good solutions and discarding certain areas.

The Ant Colony Optimization (ACO) algorithm was therefore devised. This involves the creation of an underlying graph where artificial ants can propagate and lay pheromones, a mechanism which allows the local encoding of global information, as discussed in [8–11]. The representation of the problem is the following: each node of the graph represents a technology, and two nodes are connected by an edge if, and only if, they are compatible. If an edge is present, the ant can step from a node to the next and this walk translates into a building process, where at each step, a technology is added to the configuration completed so far. Before adding a new technology, the ant must check if it is compatible with all the technologies “visited” beforehand. To this aim, the ant needs to keep an updated incompatibility list, which contains all technologies incompatible with those previously visited. To each ant step corresponds the addition of a technology to form a configuration with one more object. As the new configuration is found, it is added to the ant’s set of configurations found so far, which is then composed by a collection of configurations of increasing size. The ant’s walk ends when there are no more nodes to visit, that is, when no further technology can be added to the last configuration found. At this point, all the configurations found are evaluated against those stored in the best solution found so far. If they are suboptimal, some Local Search (LS) might be attempted (generally, if the configuration size is between 6 and 12). Pheromones are updated by a uniform evaporation and a deposition, for each configuration, on the edges visited by the ant, proportionally to the quality of the configuration. The quality criterion of a configuration is its overall cost efficiency, i.e., the total CO₂ reduction divided by the total cost. Otherwise, if the configuration is dominating or Pareto with respect to one or more stored in the best solution, pheromones are boosted (usually by a factor 1000) and proportionally to the quality of the configuration found. The configuration(s) of the best solution, dominated by the newly found configurations, if any, are identified and removed, and the better configuration (s) found, if any, are added to the best solution. In this way, the best solution keeps improving and stores the best configurations found so far by any ant.

3.1. Technical Description

All data pertaining to technologies and costs are fed into the system. The specific instance of the optimization problem is determined by choosing one value for each of the initialization parameters, i.e., year, powertrain, vehicle segment, drive cycle, cost scenario, cost type, and whether to include off-cycle technologies (see the list of parameter values in Section 5 for more detail). These values determine which technologies are present, their cost, their effective CO₂ reduction, and their incompatibility list. With these elements, since each technology constitutes a node, the underlying graph can be constructed at this stage.

As noted before, we denote the cost efficiency of a technology its CO₂ reduction per unit cost, briefly called technology efficiency (or node efficiency), as compared to the total CO₂ reduction R , divided by the total cost C , called the ‘configuration efficiency’. Certainly, using the configuration efficiency ($CE = R/C$) instead of a Pareto selection criteria on the two objectives would have resulted in optimized configurations as well, and namely on the CE itself. However, we wanted to explicitly require an improvement in each of the two objectives separately, as in criteria in Equation (5), since the aim was to have a set of configurations to choose from with separate criteria or objectives. For example, finding which configurations can reach a certain CO₂ reduction level and the expected impact of the ambition of certain CO₂ targets on actual costs.

The ACO algorithm proceeds as follows:

3.2. Solution Construction

- (1) An ant a is created and placed in the first node t_i , the first technology added to the configuration: the first node is chosen randomly in 10% of the cases and probabilistically otherwise, based on its efficiency. The ant’s list of incompatible technologies is updated with the list of incompatible

technologies of this node. The node t_i to-do-list is set “done” so it will not be added again and add the configuration to the solution the ant is constructing.

- (2) The next node t_j is determined by randomly selecting the decision criterion characterizing the ant’s inclination of exploration versus exploitation. This is done by drawing a random number to determine one of the following 3 possible decision criteria:
 - (a) Greedy (usually set to 20% of cases): go to node t_j with the maximum $p(t_j) \propto \tau(t_i, t_j) \times e(t_j)$ where $\tau(t_i, t_j)$ is the pheromone on the edge (t_i, t_j) , and $e(t_j)$ is the efficiency of t_j ; this is favoring exploitation and following the colony.
 - (b) Probabilistic (usually set to ~80% of cases): from node t_i chose next node t_j with a probability $p(t_j)$ as given above; this criteria provides a mix of exploitation and exploration.
 - (c) Random (usually set to 1% of cases): the ant goes to the next node t_j randomly; this represents pure exploration/trial.

Then to determine t_j , a second random number is drawn in the case of criteria b. or c., while if criterion a. is selected, t_j is already determined, by $\text{argmax}(p(t_j))$.

- (3) The step is made to node t_j , i.e., setting $t_i = t_j$, the node to-do list to “done” and updating the incompatibilities list with the technologies incompatible with t_j . The new configuration found is stored in the solution.
- (4) Steps 2 and 3 are repeated until no further nodes are available to be visited (i.e., not visited yet and compatible with those already present in the configuration). With the completion of the above, a solution has been constructed. The elements of this new solution are the configurations $\{T_{1a}, \dots, T_{na}\}$ each of increasing size, where subindex a denotes the ant a . These are compared one by one with those stored in the best solution found, to check if any new best or Pareto configurations can be added to the best solution. The criteria for “new best” between 2 configurations is given by the conditions (3) and (4) described in Section 2. If any configuration is dominant or Pareto, it is added to the best solution, eliminating from the best solution those configurations that are dominated by that configuration. At this point the pheromones are updated, in the following way (update of the pheromones):

- (a) Evaporation: all the pheromones on the graph undergo a uniform evaporation, i.e., all are decreased by a constant percentage (usually 20%) by multiplying by a constant ρ less than one; pheromones are never allowed to drop below a minimum level of 0.1 to avoid entering into stagnation,
- (b) Deposition: first determine if a configuration is a new optimal or not (either dominating or Pareto). If it is not, LS can be applied, as will be discussed shortly. Pheromones on the edges traversed by the ant are incremented proportionally to the configuration quality, as follows:
 - i. if not an optimal configuration: increase by $\varepsilon = S_F \times R/C$ (R and C given by Equations (1) and (2)), and with a S_F is a scale factor, depending on the problem size and costs involved.
 - ii. if a better or Pareto configuration is found, increase by $\varepsilon = S_F \times B \times R/C$, where B is the boosting factor, in the order of 10^3 .

Note that since each configuration is a combination of technologies (e.g., the order of the technologies does not matter), a deposition will be also be performed on all edges between all technologies contained. For example, if we consider the configuration $\{t_3, t_{23}, t_{54}, t_{65}\}$ then the pheromones on edges (t_3, t_{23}) , (t_3, t_{54}) , (t_3, t_{65}) , (t_{23}, t_{54}) , (t_{23}, t_{65}) , (t_{54}, t_{65}) and their reverse edges undergo i or ii above.

- (5) Repeat the process of 1 to 4. For a new ant, until a predetermined computation time has been reached. If long runs are used, and no improved configuration has been found for a certain number of iterations (usually 1 million), a boost as in 4)b.ii. is induced in the system on the best solution found so far, so that the colony is repositioned to search in that neighborhood. If this is the case, the algorithm starts some boosting cycles, typically a maximum of 10^4 cycles or until an improvement has been found, by artificially enhancing the pheromones around the best solution. This value guarantees that ants would propagate around pheromone trails of the best solution; this process is reminiscent of the elitist ants, but it is much more aggressive, and proved to contribute significantly to the long-term search. The values used were tested over several runs.
- (6) The run is finalized by outputting the best solution found, including a comparison if a benchmark solution for that problem is present. Various log files are also generated.

The schema of the ACO + LS algorithm is reported in pseudo code in Algorithm 1, which also calls the procedures of ant propagation, LS and pheromones update, reported in Algorithm from 2 to 4 below. The values of the ant parameters (e.g., the proportions of greedy, probabilistic and random steps, as well as the proportion of initial random step, ρ , S_F and B) given above were chosen in the algorithm tuning phase, where several tests were conducted to obtain the highest quality solutions in the shortest time.

Algorithm 1. ACO+LS algorithm

```

1  Initialize the graph G of nodes {ti} (technologies) with the parameters given.
2  while (keepLooping)
3      • Sa = a.propagation() which provides the solution Sa={T1a, ..., Tna}
4      • evaporate pheromones: Graph.evaporatePheromones(0.8)
5      • compare Sa to Sbest={T1best, ... Tnbest}
6          for i=1a, ..., na,
7              for j=1best, ..., nbest
8                  • if Ti dominates Tj:
9                      ○ remove Tj from Sbest and add Ti (once only)
10                 • else if Ti is Pareto with Tj, add Ti to Sbest (def. in Equation (4))
11                 • else if LSmin ≤ size(Ti) ≤ LSmax apply Local Search:
12                     ○ LocalSearch.switchTechs(Ti, 2)
13                     (LSmin, LSmax are the min and max size for LS, typically 6 and
14                     12 respectively)
15                     ○ check if Ti is better or Pareto after LS.
16                 if Ti is better or Pareto, ε = SF * B * R/C
17                 else ε = SF * R/C
18                 graph.incrementPheromones(Ti, ε)
19                 • if no improvement is found after Nit iterations:
20                     increment pheromones on Sbest={T1best, ... Tn} for M times or till an improved
21                     configuration is found (typically ε=B*R/C Nit=106 and M =104)
22                 • if (time > maxComputationTime)
23                     break
24                 else iter++
25 Finalization:
26                 • lexicographic ordering the technology IDs in each Tib of Sbest,
27                 • comparison of Sbest with benchmark if present
28
29 write logs, computation times, and best solution.
```

The ant algorithm, ant propagation and pheromone update mechanisms presented in this study are typical of those formulated in the ACO-related literature. It is worth noting that the pheromones are node based given that no ordering of the technologies in a configuration is needed. The ant choice criteria have been integrated here with the problem specific objectives, which drive the immediate reward. Also Local Search algorithm is bespoke, having been developed and integrated for this specific problem. The novelty is then how the ACO and LS have been applied to the specific problem.

Regarding the Local Search, three different methods of removing the technologies were attempted: removing randomly, removing the least efficient technologies and removing those with the largest

incompatibility list. After a few tests, substituting technologies only on the basis of efficiency appeared to be the best criteria. Once the technologies have been removed, the same number of technologies (compatible with the reduced configuration) are randomly chosen and inserted. The modified configuration is then compared to the original and, if it is better, exchanged with the previous one in the ant's solution.

Algorithm 2. Ant propagation

```

1  (1) Create an ant a and place in the first node  $t_i$ :
2      a. Generate a random number r:
3          if( $r < 0.1$ ) chose  $t_i$  randomly
4          else chose  $t_i$  probabilistically as  $e_i$  (cost efficiency)
5      b. Update the ant's list of incompatible nodes (technologies)
6      c. Set the state of this node to completed
7      d. add the configuration  $T_{1_a} = \{t_i\}$  to  $S_a$ :  $S_a = \{T_{1_a}\}$ 
8  (2) Let ant step to the next feasible node  $t_j$ :
9      a. Strategy to step: generate a random number r
10     if( $r < 0.2$ )
11         Greedy step: pick  $t_j$  such that:
12              $j = \text{argmax}(p(t_j))$  where  $p_j = \tau_{ij} * e_j$ 
13     else if ( $r < 0.99$ )
14         Probabilistic step: pick  $t_j$  distributed as:
15              $p_j = \tau_{ij} * e_j$ 
16     else
17         Random step: pick  $t_j$  randomly.
18  (3) Once the node  $t_j$  is chosen:
19      a. Move the ant in node  $t_j$ ,  $t_i = t_j$ 
20      b. update the ant incompatible node list with the nodes incompatible with  $t_j$ .
21      c. update the to-do list for  $t_j$  to done.
22      d. Store the new configuration found  $T_{n_a} = \{T_{(n-1)_a}, t_j\}$  in the solution,
23          $S_a = \{T_{1_a}, \dots, T_{n_a}\}$ 
24  (4) Repeat 2. and 3. until no further node can be visited (added) to  $T_n$  (all
25     compatible nodes have been completed/visited)

```

Algorithm 3. Local Search

```

1  • Pick the ns nodes with lowest cost efficiency and remove them from  $T_k$ :
2  for  $r=1, \dots, ns$ ,  $T_r = T_k - \{t_{r1}, \dots, t_r\}$  (typically  $n_{LS}=1$  or 2)
3  • Update the incompatibility list  $\lambda_r$  for the reduced configuration  $T_r$ 
4  • repeat ns times:
5      ■ Among the compatible nodes with  $\lambda_r$  pick one randomly and insert it in  $T_r$ 
6      ■ Update  $\lambda_r$ 

```

Algorithm 4. Pheromones update

```

1  • Evaporation:  $\forall i, j$ , with  $i \neq j$ :  $\tau_{ij} = \max(\rho * \tau_{ij}, 0.1)$ 
2  where  $\rho$  is the persistency constant ( $0 < \rho < 1$ , typically  $\rho = 0.8$ )
3
4  • Deposition: for  $i, j$  such that  $t_i$  and  $t_j \in T_k$ :  $\tau_{ij} = \tau_{ij} + \epsilon$ 
5  where  $\epsilon$  is the deposition level ( $\epsilon > 0$ )

```

The Local Search is typically run for intermediate configurations sizes, usually of 6 to 12 technologies, since this is a greatly populated area in the solution space. It is performed only if the configuration is not an optimal one. It proceeds by removing one or more technologies from some chosen configuration and re-calculating the incompatibility list without the removed technologies; finally,

some alternative technologies, compatible with the reduced configuration, are added. The number of technologies to switch can be chosen by the user, but as it heavily affects the efficiency, it was typically set to 2.

Finally, we note that many tests have been conducted in order to identify reasonable values for the various algorithm parameters over a wide set of problem instances which guarantee convergence to high quality solutions in timeframes on the order of 10 min. Detailed discussion of these tests is beyond the scope of this paper.

4. Results and Benchmarking

The ACO + LS algorithm presented in the previous section, was tested for a wide range of optimization scenarios and employed to calculate optimal solutions, which are the inputs for the calculation of the vehicle CO₂ emission reduction cost curves, as introduced in [21]. For cars and vans this included optimizations for all powertrains and vehicle segments, with and without off-cycle technologies, for 2025 and 2030 under the Worldwide harmonized Light vehicles Test Procedure (WLTP) test cycle (i.e., 224 optimization runs), and a number of other cases, as presented in Section 5 and in [22,23]. Moreover, the tool was employed for a number of additional computations. Each optimization was run for 10 min, yielding high quality solutions for the given algorithm parameters. By performing these runs in parallel, it was possible to complete overnight (14 h run-time) even a larger set of runs, up to 2000, taking advantage of 32 processors-threads of the Intel® Xeon® CPU E5-2687W at 3.10 GHz.

In an earlier approach, a heuristic method was applied for assessing vehicle CO₂ reduction technologies and costs as presented in [24]. In order to solve the problem, a cloud of data points (i.e., tuples of CO₂ emission reduction and cost values) was created from all possible combinations of different CO₂ reduction technologies. This method was able to generate results for problems up to about 30 technologies. However, it proved to be inefficient for larger and constrained problems, so to handle these, a more efficient heuristic was developed, at first utilising a procedure where the configurations are determined iteratively by using a set of constraints based on incompatible technologies, with a Genetic Algorithm (GA) to create and select optimized solutions.

Genetic Algorithms were proved to effectively solve a number of optimization problems and also in particular the MKP, as reported in [25]. In recent years, such algorithms have also been developed and made available in MATLAB® as a toolbox [26].

Since the current problem has two objectives, a possible approach with a GA is to divide the cost range into a certain number of 'slices' (usually a few hundred, from a minimum to maximum pre-determined cost range). In each cost slice, the algorithm seeks configurations with the maximum CO₂ reduction value, taking into account the compatibility constraints. The problem is thus reduced to a single objective optimization in each cost slice. In a preliminary study, it appeared crucial to establish an appropriate number of these slices since it heavily influences the computational time. The authors reported, after running some tests on an Intel Core i5-3340M CPU at 2.70 GHz, that a value of 200 slices yielded a reasonable computation time of 4-5 min per instance, as compared to 100 slices (about 2 min), 400 slices (about 10 min) and 1000 slices (about 30 min). This choice was also made since further increasing the number of slices, was not adding a sufficient improvement to justify the increased computational times.

The GA approach demonstrated to be able to handle problems of up to about 50 technologies. The computational work was carried out in 2015 using MATLAB® R2013a, using the function *ga* which is contained within the MATLAB® Global Optimization Toolbox to launch the optimization [27]. This function, if not specified by the user, automatically adjusts the levels of mutation and cross-over for the constrained optimization problem, as the total cost of a configuration must lie in the cost slice. The slices are created by computing, for each case, the minimum and maximum cost. The former is the minimum cost among the technologies considered in the instance problem, while the latter is the sum of the cost of all the technologies. The schema of the algorithm is given in Algorithm 5.

Although this approach is quite efficient, as the number of CO₂ reduction technologies start to increase, it becomes computationally more and more inefficient and time consuming. We recall that the search space grows very quickly, as 2^N , where N is the number of technologies.

Algorithm 5. Genetic Algorithm

```

1  Given the problem with N technologies:
2  • Create n cost slices  $s_i$ : if  $\Delta=(C_{\max}-C_{\min})/n$   $s_i=[C_{\min}+i*\Delta, C_{\min}+(i+1)*\Delta]$   $i=0, \dots, n-1$ 
3  • Repeat over each slice  $s_i$  the call to ga, the Genetic Algorithm, with the following
4  parameters:
5       $[x, \text{CO}_2 \text{ reduction}] = \text{ga}(f, n, A, b, [], [], lb, ub, [], \text{IntCon})$ 
6
7  where  $f$  is the product of used technology cost and CO2,  $N$  is the number of unique
8  technologies,  $A$  and  $b$  form the linear constraints  $A*x \leq b$  and encodes
9  technologies incompatibilities,  $lb$  and  $ub$  are vectors of lower and upper bounds;
10  $\text{ga}$  enforces that iterations stay above  $lb$  and below  $ub$ ,  $\text{IntCon}$ : vector of positive
11 integers taking values from 1 to  $N$ . Each value in  $\text{IntCon}$  represents an  $x$  component
12 that is integer-valued.
13 • Output the results: one point (a configuration) per each slide  $s_i$ 

```

In the following section, ACO + LS outcomes have been compared against the outcomes achieved with the GA approach. The GA runs are the benchmark runs, i.e., they were not repeated for each problem instance, but rather each is the best solution found over all the runs made for that instance; for some instances different slice resolutions were ran, with different times to improve solution quality. We have reported and compared with the best solution found for a particular instance.

Figure 1 shows an example of the CO₂ reduction/cost-tuples resulting from both approaches, run for a typical case. The coloured (or grey) dots represent ACO + LS solutions, where the colours (or grey shade) indicate the size of the configuration. The black squares represent the GA benchmark solutions.

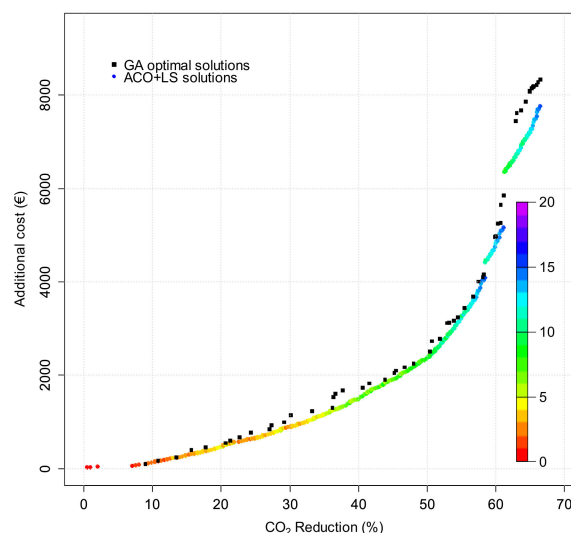


Figure 1. Comparison of CO₂ reduction and cost for optimal configurations found by GA and ACO + LS. Black squares represent GA configurations while colored (or grey) dots represent ACO + LS configurations. The color (or grey shade) represent the configuration size).

For this case, we note that there are a total of 76 Pareto optimal GA configurations (points), against the 287 optimal configurations found with ACO + LS. Moreover the GA configurations are almost always dominated by the configurations found by ACO + LS. The high number of solutions found by the ACO + LS makes the Pareto front more densely populated, which is desirable when searching differentiation or alternative configurations.

We note also a side effects of the GA slicing: the best solution found in a slice, can be dominated by solutions in other slices. In other words, dividing the 2nd objective by M slices, does not guarantee that the Pareto front will have M points. In this case, pareto-optimal configurations are not found in some slices (as their existence can be seen by the ACO + LS), therefore the GA algorithm converges to sub-optimal configurations in a number of cases.

The two noticeable discontinuities are due to two high-cost technologies, usually characterized also by a high CO₂ reduction; these, once added to a lower size optimal configuration, result in sudden increase of the additional cost and CO₂ reduction, thus appearing as a discontinuity of the pareto front.

For a more systematic comparison, ten problems were chosen, as listed in Table 1, with different initial parameters for year, powertrain, vehicle segment and test cycle, where GA benchmark solutions were available. For Problems 2, 8 and 9, the GA was run over 1000 iterations (1 h runs).

Table 1. Reference table of the benchmark problems, with typical cost scenario and total cost type. See Nomenclature for abbreviations or Section 5 for extended descriptions of the parameter values.

Problem Number	Year	Powertrain	Segment	Cycle	Include Off-Cycle
Problem 1	2015	SI ICE + HEV	Lower Medium Car	NEDC	FALSE
Problem 2	2025	SI ICE + HEV	Lower Medium Car	WLTP	FALSE
Problem 3	2015	CI ICE + HEV	Lower Medium Car	WLTP	FALSE
Problem 4	2020	CI ICE + HEV	Lower Medium Car	NEDC	FALSE
Problem 5	2025	CI ICE + HEV	Lower Medium Car	NEDC	FALSE
Problem 6	2015	CI ICE + HEV	Lower Medium Car	NEDC	FALSE
Problem 7	2015	CI ICE + HEV	Upper Medium Car	WLTP	FALSE
Problem 8	2025	CI ICE + HEV	Lower Medium Car	WLTP	TRUE
Problem 9	2025	SI ICE + HEV	Small Car	WLTP	TRUE
Problem 10	2025	BEV	Lower Medium Car	WLTP	TRUE

For each problem, ten repeated ACO + LS runs were made with a runtime of 10 min and 1 h, respectively. Then ACO + LS solutions (sets of all configurations found) were compared to GA benchmark solutions in the following way.

Accordingly, to the optimality criteria defined in Equation (3), a configuration of a solution is dominated if there is at least one configuration of the second solution that is dominating it. Therefore, to have an appropriate comparison of two Pareto optimal fronts, also called best solutions found or simply solutions, S_1 and S_2 , each configuration T_k of the solution S_1 has to be compared to the set of configurations of the solution S_2 . An evaluation is made to check if T_k is dominating, dominated, additional Pareto for S_2 or equal (exactly the same cost and CO₂ reduction, with 2 precision digits for costs and 8 precision digits for CO₂ reduction - these values were set considering the precision of the GA benchmark configurations at hand). If it is dominating, the number of dominated configurations of S_2 is estimated.

Then the reverse is done, to evaluate, one by one, how many configurations of S_2 are dominating, dominated, additional Pareto, or equal with respect to S_1 . Since ACO + LS runs were repeated 10 times for each problem, for each run, the number of dominating, additional Pareto, equal and dominated configurations were computed for each run. Finally, their averages were estimated, along with their 95% confidence interval over the repeated runs. The runs of 10 min and 1 h are reported in Tables 2 and 3, respectively.

The tables contain the comparison of the GA benchmark solutions S_{GA} versus ACO + LS solutions S_{ACO+LS} (first five columns), and then the reverse (last five columns). In particular, the size of the benchmark solutions is reported (in the tables, the column “configs. found”, column 1), which is measure of the search efficiency of the respective algorithm: the greater the number of configurations, the greater is their diversity and thus the possibility of combing technologies/objects in different ways. It is also reported, for each problem, the mean number of configurations of S_{GA} that are better than S_{ACO+LS} , and the mean number of configurations of S_{ACO+LS} that are dominated by the S_{GA} . Then, in the following columns 4 and 5, the mean number of additional Pareto configurations, and the mean number of equal configurations are reported (equal configurations is the only quantity which is invariant when comparing S_1 to S_2 or vice-versa). In the second part of the tables, columns 6–10, the reverse comparison is made, for the configurations of S_{ACO+LS} versus S_{GA} .

In the tables, it can be seen that the GA benchmark solutions contain additional optimal configurations in just a few cases to the optimal configurations found by ACO + LS (in the tables, these cases are marked in bold).

A value of 0.10 in column 2, means that, over 10 repeated runs, one GA configuration was better than some ACO + LS configuration (s); the value of 0.10 in column 3, tells that only one ACO + LS configuration was dominated by that GA configuration. In some other cases, a few additional Pareto configurations were found by GA. Finally a few other configurations are the same exact configurations as found by ACO + LS (the columns 5 and 10 are indeed the same). Most relevantly, in most cases ACO + LS configurations dominate the GA configurations (columns 7 and 8) or are additional Pareto (column 9) to those. The number of optimal configurations found by ACO+LS and GA is reported in columns 1 and 6 respectively, and the different order of magnitude of the two is noticeable. As computation time is increased from Table 2 to Table 3, the ACO + LS algorithm finds a slightly higher number of optimal configurations and convergence is increased over the ten repeated runs, as shown by the consistent decrease of the 95% confidence interval. Also the number of the ACO + LS dominated configurations shows a clear tendency to diminish further, as well as the GA additional Pareto solutions to ACO + LS.

It is possible to run the ACO + LS algorithm for many hours in an attempt to find even better solutions, or repeat the same runs to see how consistent is the convergence to the final solution. On an Intel Xenon at 3.10 GHz, it was found that approximately 10 min is suitable to find an optimized solution using ACO + LS. Given the high number of configurations found, a substantial improvement with respect to the GA is achieved. Depending on the number of available technologies of the problem instance, the ACO + LS algorithm typically completes between 30 and 60 million iterations (ants) in 10 min.

Besides the 10 problems discussed above, an indirect comparison was made between the fits found by the new approach for several thousand input datasets and cost scenarios. For each, the optimal configurations are fitted with a Levenberg-Marquardt non-linear regression algorithm using the *nlsLM* function of the *minpack.lm* library in R [28]. This provides a so-called cost function, the analytical form of the Pareto front, as documented and used in [22,23], which are then used for various other applications. This extensive exercise enhanced the authors’ confidence in the efficiency and robustness of the ACO + LS algorithm.

Finally, as a further evaluation of the ACO + LS method, we examined the cloud of points in the CO₂ reduction/cost plane pertaining to the configurations the ants visited during their search. By inspecting the cloud, the ant colony’s distance to the Pareto front can be evaluated. If the colony stays close enough to the optimal solution, it has a higher likelihood of finding improved configurations and avoiding stagnation in local minima. The search process is driven by the parameters which determine the ants’ behavior, particularly exploitation versus exploration settings (the relative share between greedy, probabilistic and random choices during propagation). The following Figure 2 shows this cloud for Problem 1. Each of the small points relates to a configuration found by the ants during their search. The points are a random extract of 5000 points over the last part of the run; only this small

number of points is extracted for visualization purposes, since the total number of points are several millions, for a typical run. Similar plots were also made for initial or intermediate intervals of the run, to monitor the activities of ants in the colony.

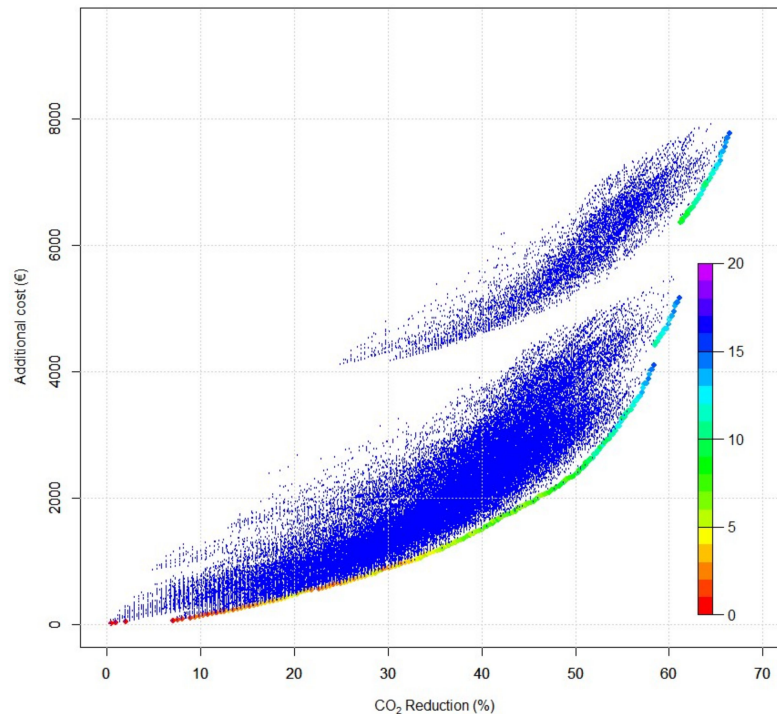


Figure 2. Randomly sampled intermediate configurations visited by the ants (small blue dots) along with optimal configurations found by ACO (colored or grey shades-bigger dots of the Pareto front).

The most significant ant parameters were varied and studied. We found that for runs of 10 min or greater, there were no significant differences in the colony behavior between the first and last fifth of iterations once the algorithm is properly tuned. This means that 10 min are sufficient for the ant colony to stabilize and indicates that it is likely that even less time would be required to find optimal solutions for the problems considered. In Figure 2 the optimal configurations (the Pareto front) found by ACO + LS are plotted indicating the number of technologies present in the particular configuration. Ideally, the cloud generated by ACO + LS should stay as close as possible to the optimal solution (which also changes during the optimization process) such that small variations in the ants' paths (explorations) can make them find new optimal configurations.

We note that configurations of intermediate size, i.e., 6 to 12 technologies, constitute the most populated area. Figure 2 visually shows why it was decided to run LS only on points in this region: to enhance exploitation in this highly populated area of the search space. Mathematically, this is due to the summation term in (6), where combinations are at their maximum when $n = N/2$.

In summary, the ACO + LS approach has been shown to be able to handle the computational complexity of the Technology Packaging Problem quite efficiently, in particular with the following benefits:

- (1) It is highly efficient: it provides nearly optimal solutions within a few minutes of computation, and high-density solutions within 10 to 15 min. For the problems examined, with appropriate settings of the algorithm parameters, the ACO + LS completes roughly one million iterations in less than 10 min, reaching a considerable diversity in the configurations.
- (2) It does not require reduction to a single objective function like slicing, or other ad-hoc or a priori operations, and can potentially deal with more than two objectives.

- (3) It finds highly populated sets near the Pareto optimal front - that is, a large number of nearly optimal configurations which might represent an alternative when different configurations might be needed (a manufacturer might have already certain packages in place) in terms of CO₂ reduction and cost.
- (4) It supports 2nd order corrections of CO₂ reduction diminishment due to technologies overlapping or interaction. This aspect has not yet been deployed.

In particular, as a consequence of (3) above, the so-called technology pathways can be identified, i.e., certain points or regions of the Pareto front can be associated with different configurations that reach them in the most cost-efficient way from sub-optimal configurations, just by adding specific technologies. In other words, it is possible to gather concrete indications as to what are the most suitable technologies to add to pre-existing packages to achieve given CO₂ reduction targets at a minimum cost, so that the transitions from these packages to new optimized ones can be done with ease and continuity.

We finally note that due to the novelty of the problem, we could not compare with other algorithms in the literature. This study aimed to present a method that could solve this problem most efficiently and with high quality solutions. This method could constitute a benchmark for future algorithms to solve this problem, and certainly provides clear indications that it is exploring extensively and efficiently the search space. The same methodology was later applied with minor changes to provide input for discussing CO₂ reduction technologies for heavy duty vehicles, which demonstrates the flexibility of the method.

5. Applications

We applied the method to identify optimal technology packages for reducing CO₂ emissions from light duty vehicles (LDVs, i.e., passenger cars and vans) [22] and later also for the heavy duty vehicles (HDVs) [23]. The approach presented in this study was employed within the analytic work supporting the impact assessments for post-2020 LDV and HDV CO₂ standards in the EU.

A dataset of more than 80 LDV CO₂ emission reduction technologies, including their reduction potentials, costs and mutual compatibilities was provided in [29]. Similarly, an extensive study was carried out for HDVs with data in [30]. For each problem, once the optimal configurations are found with the ACO + LS, a so-called cost curve can be found by fitting, and it describes the Pareto front of costs versus CO₂ reductions in a continuous analytical form.

A number of values for the initialization parameters exist, which define a specific instance of the optimization problem. For the LDVs these are namely (a list of abbreviations is also available in Nomenclature Section):

- Year, with three different values: 2020, 2025, 2030. In some cases 2015 was also used.
- Vehicle powertrain, typically with eight distinct values:
 - Spark Ignition (i.e., gasoline) Internal Combustion Engine and Hybrid (SI ICE + HEV)
 - Compression Ignition (i.e., diesel) Internal Combustion Engine and Hybrid (CI ICE + HEV)
 - Spark Ignition Plug-in Hybrid (SI PHEV)
 - Compression Ignition Plug-in Hybrid (CI PHEV)
 - Spark Ignition Range-Extended Electric Vehicle (SI REEV)
 - Compression Ignition Range-Extended Electric Vehicle (CI REEV)
 - Battery Electric Vehicle (BEV)
 - Fuel Cell Electric Vehicle (FCEV)
- Vehicle segment, with seven values: Small Car, Lower Medium Car, Upper Medium Car, Large Car, Small Van, Medium Van, Large Van.
- Drive cycle within which CO₂ emissions are determined, with three values:

- New European Drive Cycle (NEDC)
- Worldwide harmonized Light vehicles Test Procedure (WLTP)
- Real World Drive Cycle (RDC)
- Cost scenario reflecting the rate of cost reduction over time for a technology through learning, with three values: High, Low, Typical.
- Cost type: with typically two values:
 - Direct cost, i.e., manufacturing costs
 - Total costs, i.e., direct costs plus indirect costs, where indirect costs represent additional costs related to a technology such as R&D, warranties, marketing, etc.
- Off-cycle technologies: a number of CO₂ emission reduction technologies cannot be measured under a type approval drive cycle such as NEDC or WLTP, e.g., led lights, solar roofs or tire pressure monitoring. If off-cycle technologies flag is set to 'true', these are included in the set to optimize over, otherwise they are excluded.

The possible combinations of all these parameters give rise to a total combination of 6048 different instances. For the sake of most of applications, we set the cost type equal to total cost, which results in a total of 3024 possible optimization problem instances to be run for each scenario. This is quite a computational effort: if ran in sequence they correspond to about 21-day computation time for 10 min runs, or 7 days if only interested in WLTP cycle.

It is to be noted that the set of available CO₂ reduction technologies are updated infrequently, depending on the availability of new data on technologies and their costs as well as on policy needs. Therefore, the addition of new technologies doesn't constitute any computational issue, as well as the variation of their costs, having been taken into consideration via 3 technology cost scenarios in the construction of the Pareto front for modelling the Cost Curves.

Cost Curve Construction

On the basis of the optimal configurations found by the ACO optimization described in the previous sections, vehicle emission reduction cost curves can be constructed to interpolate analytically these data points. The construction of cost curves consists in the fitting of the Pareto front configurations/points. This fitting procedure was a considerable part of the work, and it is worth noticing that all the Pareto fronts were fitted with an analytical form deriving by one single generalized functional form, with at most 4 fitting parameters. The cost curves are an essential input for applications such as the evaluation of different CO₂ reduction scenarios, e.g., for calculating the costs associated with a certain desired CO₂ standard for vehicles, identifying cost-minimizing distributions of CO₂ reduction efforts across different vehicle types and segments.

A number of post-processing steps had to be applied to the raw output from the optimization procedure, to take into account some necessary adjustments. The post-processing steps, the cost curve fitting procedure and the resulting cost curves are documented in [22] for LDVs, and in [23] for HDVs.

We have conducted extensive computational campaigns to address a number of variations and hypotheses for scenario analysis, considering variations of the original problem instances where for example a certain technology is desired to be always present, or in another case, where a certain technology could be considered to be less effective (in term of CO₂ reduction) than expected or some other more costly, to derive the optimal configurations and related cost curves. These results, obtained in relative short time scales, were made possible thanks to the properties of fast convergence and efficiency of the ACO + LS methodology.

6. Conclusions

We have presented a method based on a metaheuristic with Ant Colony Optimization combined with a Local Search algorithm to efficiently solve the Technology Packaging Problem, of finding feasible configurations of CO₂ reduction technologies, maximizing the CO₂ reduction and minimizing costs.

The results presented show that this methodology provides solutions of increased quality and size versus other approaches, e.g., genetic algorithms, thus offering the possibility of finding improved configurations in a highly efficient and easily adaptable manner. This has enhanced the capability to explore optimal strategies under increasingly numerous and complex technology choices against various CO₂ reduction targets. The method can easily accommodate more technology options or perform analyses of different technology pathways. As the runtime is greatly improved, it has been employed to run multiple scenario analyses, including e.g., variations in technology impact, costs, and test cycles.

This method has been employed for supporting the impact assessments for post-2020 CO₂ standards for cars, vans and trucks in the EU, considering diverse road vehicle fleet compositions. Different scenarios for the post-2020 EU CO₂ emission standards have been evaluated by constructing vehicle emission reduction cost curves as described in the present paper, and, building on them, identifying an optimal distribution of efforts among different powertrains and segments which compose the fleet, calculating additional manufacturing costs as well as fuel savings, and computing total additional costs or savings from emission reduction. Further to applications in policy support, this methodology could also be applied by the automotive industry itself.

Author Contributions: Conceptualization: A.V.D., J.K. and C.T.; methodology: A.V.D., B.W. and N.H.; software: A.V.D., B.W. (for GA); validation: J.K. and C.T.; formal analysis: A.V.D.; investigation: N.H.; data curation: N.H. and B.W.; writing—original draft preparation: A.V.D.; writing—review and editing: all authors; visualization: A.V.D.; supervision: J.K. and C.T.; project administration: C.T.; funding acquisition: N.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported through the Horizon 2020 direct funding of the Joint Research Centre (JRC) of the European Commission, for the JRC authors. The work of the other authors was funded by the European Commission under the framework contract CLIMA.C.2/FRA/2012/0006. The views expressed are purely those of the authors and may not in any circumstances be regarded as stating an official position of the European Commission.

Acknowledgments: The authors wish to thank Yannis Drossinos (JRC), for the valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature: List of Abbreviations

BEV	Battery Electric Vehicle
CI ICE + HEV	Compression Ignition (i.e., diesel) Internal Combustion Engine and Hybrid
CI PHEV	Compression Ignition Plug-in Hybrid
CI REEV	Compression Ignition Range-Extended Electric Vehicle
FCEV	Fuel Cell Electric Vehicle
HDV	Heavy Duty Vehicle (truck)
LDV	Light Duty Vehicle (car or van)
LS	Local Search
NEDC	New European Drive Cycle
RDC	Real World Drive Cycle
SI ICE + HEV	Spark Ignition (i.e., gasoline) Internal Combustion Engine and Hybrid
SI PHEV	Spark Ignition Plug-in Hybrid
SI REEV	Spark Ignition Range-Extended Electric Vehicle
WLTP	Worldwide harmonized Light vehicles Test Procedure

References

1. Martello, S.; Toth, P. *Knapsack Problems: Algorithms and Computer Implementations*; Wiley: New York, NY, USA, 1990.
2. Kellerer, H.; Pferschy, U.; Pisinger, D. *Knapsack Problems*; Springer: Berlin, Germany, 2004.
3. Lust, T.; Teghem, J. Multiobjective Multidimensional Knapsack Problem: A survey and a new approach. *Int. Trans. Oper. Res.* **2012**, *19*, 495–520. [[CrossRef](#)]
4. Rodger, J.A. Toward reducing failure risk in an integrated vehicle health maintenance system: A fuzzy multi-sensor data fusion Kalman filter approach for IVHMS. *Expert Syst. Appl.* **2012**, *39*, 9821–9836. [[CrossRef](#)]
5. Fonseca, C.M.; Fleming, P.J. Genetic algorithms for multi-objective optimization: Formulation, discussion, generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, 17–21 July 1993; pp. 416–423.
6. Srinivas, N.; Deb, K. Multi-objective optimization using non dominated sorting in genetic algorithms. *Evol. Comput.* **1994**, *2*, 221–248. [[CrossRef](#)]
7. Hajela, P.; Lin, C.-Y. Genetic search strategies in multicriterion optimal design. *Struct. Optim.* **1992**, *4*, 99–107. [[CrossRef](#)]
8. Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*; Oxford University Press: New York, NY, USA, 1999.
9. Dorigo, M.; Di Caro, G.A. Ant colony optimization: A new meta-heuristic. In Proceedings of the Congress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; pp. 1470–1477.
10. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [[CrossRef](#)] [[PubMed](#)]
11. Dorigo, M.; Gambardella, L.M. Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [[CrossRef](#)]
12. Dorigo, M.; Britteri, M.; Blum, C.; Clerc, M.; Winfield, T.S.A. *Ant Colony Optimization and Swarm Intelligence*; Springer: Berlin/Heidelberg, Germany, 2008.
13. Donati, A.V.; Montemanni, R.; Casagrande, N.; Rizzoli, A.E.; Gambardella, L.M. Time dependent vehicle routing problem with a multi ant colony system. *Eur. J. Oper. Res.* **2006**, *185*, 1174–1191. [[CrossRef](#)]
14. Montemanni, R.; Gambardella, L.M.; Rizzoli, A.E.; Donati, A.V. Ant colony system for a dynamic vehicle routing problem. *J. Comb. Optim.* **2005**, *10*, 327–343. [[CrossRef](#)]
15. Donati, A.V.; Darley, V.; Ramachandran, B. An ant-bidding algorithm for generalized flow shop scheduling problem: Optimization and Phase Transitions. In *Advances in Metaheuristics for Hard Optimization*; Springer: Berlin/Heidelberg, Germany; New York, NY, USA, 2008; pp. 111–138.
16. Rada-Vilela, J.; Chica, M.; Cordon, O.; Damas, S. A comparative study of multi-objective ant colony optimization algorithms for the time and space assembly line balancing problem. *Appl. Soft Comput.* **2013**, *13*, 4370–4382. [[CrossRef](#)]
17. Leguizamón, G.; Michalewicz, Z. A new version of ant system for subset problem. In Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), Washington, DC, USA, 6–9 July 1999.
18. Fidanova, S. Ant colony optimization and multiple knapsack problem. In *Handbook of Research on Nature Inspired Computing for Economics and Management*; Rennard, J.-P., Ed.; IGR Global: Hershey, PA, USA, 2006; Volume 2, pp. 498–509.
19. Alaya, I.; Solnon, C.; Ghéira, K. Ant colony optimization for multi-objective optimization problems. In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, Patras, Greece, 29–31 October 2007; pp. 450–457.
20. Kong, M.; Tian, P.; Kao, Y. A new ant colony optimization algorithm for the multidimensional Knapsack problem. *Comput. Oper. Res.* **2008**, *35*, 2672–2683. [[CrossRef](#)]
21. Hill, N.; Windisch, E.; Kirsch, F.; Horton, G.; Dun, C.; Hausberger, S.; Matzer, C.; Skinner, I.; Donati, A.V.; Krause, J.; et al. *Improving Understanding of Technology and Costs for CO₂ Reductions from Cars and LCVs in the Period to 2030 and Development of Cost Curves*; Service Request 4 to LDV Emissions Framework Contract; Final Report for DG Climate Action, Ref. CLIMA C.2/FRA/2012/0006; Ricardo Energy & Environment ref.: ED59621; Ricardo: Shoreham-by-Sea, UK, 2016.

22. Krause, J.; Donati, A.V.; Thiel, C. *Light Duty Vehicle CO₂ Emission Reduction Cost Curves and Cost Assessment—the DIONE Model*; EUR 28821 EN; Publications Office of the European Union: Luxembourg, 2017; ISBN 978-92-79-74136-4.
23. Krause, J.; Donati, A.V. *Heavy Duty Vehicle CO₂ Emission Reduction Cost Curves and Cost Assessment enhancement of the DIONE Model*; EUR 29284 EN; Publications Office of the European Union: Luxembourg, 2018; ISBN 978-92-79-88812-0.
24. Smokers, R.; Vermeulen, R.; Van Mieghem, R.; Gense, R.; Skinner, I.; Fergusson, M.; MacKay, E.; Brink, P.T.; Fontaras, G.; Samaras, Z. *Review and Analysis of the Reduction Potential and Costs of Technological and Other Measures to Reduce CO₂-Emissions from Passenger Cars*; TNO Final Report; TNO: Delft, The Netherlands, 2011.
25. Chu, C.; Beasley, J.E. A genetic algorithm for the multidimensional knapsack problem. *J. Heuristic* **1998**, *4*, 63–86. [[CrossRef](#)]
26. Chipperfield, J.; Fleming, P.J.; Pohlheim, H.; Fonseca, C.M. A genetic algorithm toolbox for MatLab. In Proceedings of the International Conference on Systems Engineering, Coventry, UK, 6–8 September 1994; pp. 200–207.
27. MatLab Documentation on Genetic Algorithms and ga Function. Available online: <http://it.mathworks.com/help/gads/ga.html> (accessed on 31 May 2020).
28. Elzhov, V.; Mullen, K.M.; Spiess, A.N.; Bolker, B. R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds. R Package ‘minpack.lm’, Version 1.2–1. November 2016. Available online: <https://cran.r-project.org/web/packages/minpack.lm/minpack.lm.pdf> (accessed on 31 May 2020).
29. The Dataset of CO₂ Reduction Technologies. Available online: https://ec.europa.eu/clima/sites/clima/files/transport/vehicles/docs/technology_results_web.xlsx (accessed on 31 May 2020).
30. TNO; Graz University of Technology; CE Delft; ICCT. Support for Preparation of the Impact Assessment for CO₂ Emissions Standards for Heavy Duty Vehicles. Service Request 9 under Framework Contract no CLIMA.C.2./FRA/2013/0007. Available online: https://ec.europa.eu/clima/sites/clima/files/transport/vehicles/heavy/docs/support_impact_assessment_hdv_en.pdf (accessed on 31 May 2020).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).