mathematics

MDPI

Article

Optimization of the Real-Time Response to Roadside Incidents through Heuristic and Linear Programming

Roman Buil [1,2,*], Jesica de Armas [3], Daniel Riera [1] and Sandra Orozco [2]

[1] Studies of Computer Science, Multimedia and Telecommunications, Universitat Oberta de Catalunya, Rambla Poble Nou, 18, 08018 Barcelona, Spain; drierat@uoc.edu
[2] Accenture S.L., Passeig de Sant Gervasi, 51, 08022 Barcelona, Spain; s.orozco.martin@accenture.com
[3] Department of Economics and Business, Universitat Pompeu Fabra, Ramon Trias Fargas, 25-27, 08005 Barcelona, Spain; jesica.dearmas@upf.edu
* Correspondence: roman.buil.gine@accenture.com or rbuilg@uoc.edu; Tel.: +34-667-605-170

Abstract: This paper presents a solution for a real-world roadside assistance problem. Roadside incidents can happen at any time. Depending on the type of incident, a specific resource from the roadside assistance company can be sent on site. The problem of allocating resources to these road-side incidents can be stated as a multi-objective function and a large set of constraints, including priorities and preferences, resource capacities and skills, calendars, and extra hours. The request from the client is to a have real-time response and to attempt to use only open source tools. The optimization objectives to consider are the minimization of the operational costs and the minimization of the time to arrive to each incident. In this work, an innovative approach to near-optimally solving this problem in real-time is proposed, combining a heuristic approach and linear programming. The results show the great potential of this approach: operational costs were reduced by 19%, the use of external providers was reduced to half, and the productivity of the resources owned by the client was significantly increased.

Keywords: roadside assistance; resources scheduling optimization; real-time allocation; multi-objective function

Citation: Buil, R.; de Armas, J.; Riera, D.; Orozco, S. Optimization of the Real-Time Response to Roadside Incidents through Heuristic and Linear Programming. *Mathematics* **2021**, *9*, 1982. https://doi.org/10.3390/math9161982

Academic Editor: Inmaculada Rodríguez-Martín

Received: 7 July 2021
Accepted: 13 August 2021
Published: 19 August 2021

1. Introduction

Roadside assistance is a service that helps the drivers of cars, motorcycles, and bicycles whose vehicles have suffered a mechanical failure (or accident) that has left them stranded. Resolving one of these incidents may involve starting a car, diagnosing and repairing the problem, towing a vehicle, changing a flat tire, freeing a vehicle that is stuck in the snow, or helping people who have been trapped. Depending on the type of incident, a company-specific resource will be dispatched to the site. If the incident can be repaired on the spot, a technician will provide assistance with a properly equipped vehicle; otherwise, if the incident cannot be repaired, a tow truck will most likely be used to remove the damaged vehicle.

Roadside assistance companies serving their customers in dynamic locations face the same problem: defining resource allocation that minimizes operating costs while maintaining the level of service above a certain threshold. Resources are limited and often have specific characteristics to take into account. For example, a resource may need a certain skill to qualify as a candidate to be assigned to a certain task. Different resources may also incur different fixed and/or variable costs, which must be taken into account when comparing options. Additionally, issues may also have some specific requirements or characteristics that could affect the allocation, such as their urgency or even specific requests from customers to provide them with a certain type of resource. Finally, regulations regarding worker breaks, as well as contract requirements, will determine the feasibility of a solution.

This research, far from attempting to improve the current theoretical solutions, focuses on facing a real-world practical problem, proposing and implementing a solution that takes into account all the restrictions (whatever the type) collected by the company. The Real Automòbil Club de Catalunya (RACC (https://www.racc.cat) accessed on 18 August 2021) is a roadside assistance company that operates in Spain. They cover almost 2000 mechanical incidents every day, manage around 145 resources, and make use of autonomous tow trucks as well as external providers to cover those incidents that cannot be covered by their own resources (either because the arrival time would be too late, the allocation of their own resources is too expensive, or the incident type requires a skill set that is not available at the time).

Currently, the RACC is forecasting the number of incidents that occur each day at a very high level, without distinction between different geographic areas, type of incident, or time of day. With this estimate, they can make medium to long-term decisions, such as fleet size or determining the number of resources to work each shift and scheduling vacations. However, operational and tactical decisions often require smaller granularity: having a good approximation of the number of incidents of a certain type at a specific time of day would allow them not only to improve the organization of work shifts but also to locate a resource with a skill set closer to where an assignable incident is most likely to occur. Thus far, all resources started their shifts at some fixed base locations and then moved according to the work orders they were assigned to.

The company has software that automatically assigns the incidents that are simple enough, that is, those that require skills that most of the resources have. However, this happens only 40% of the time, while the other 60% of assignments are done manually. In both cases, this decision is made primarily based on the proximity of each resource to the incident that is being assigned, with closer resources being more likely to be selected if they meet the minimum requirements.

To assess the quality of assignments, two main Key Performance Indicators (KPIs) are used: the resource utilization (the time during which a resource is traveling or working to resolve an incident out of the total working hours of the resource considered) and the compliance percentage with the Service Level Agreement (SLA). The SLA is the maximum contractual time between the moment the request is received and the moment the assigned resource arrives at the scene of the incident. If the resource arrives later than specified, a penalty is added to the operational cost of the service.

Both the current SLA and resource utilization are lower than expected. Therefore, the RACC has been looking for a solution that improves its daily operations, making better use of its own resources, reducing operating costs and complying with the SLA of its clients whenever possible. The solution designed to achieve this objective consists of the integration of three modules (See Figure 1):

- Prediction of the number of incidents: Machine Learning Model Competition to predict the incidents by region, hour, and type. (Module 1 in Figure 1)
- Location Optimization Algorithm: Dynamic decision making of the optimal location of each resource based on the data history, clusters, and current status. (Module 2 in Figure 1)
- Real-Time Resource Allocation Optimization Algorithm: To decide which resource should be allocated to each incident in real-time, taking into account all business rules and aiming to minimize operational costs while ensuring the target service level. (Module 3 in Figure 1)

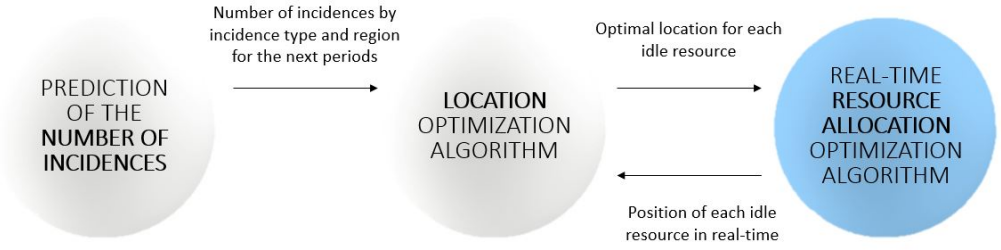

Figure 1. Relationship between the three modules (in blue, scope of this paper).

The third module is the one that has the most direct impact on day-to-day operations and, therefore, the one that is most likely to lead to a significant reduction in operating costs if optimized. For this reason, the RACC decided to take this as a first step, to improve it in the future by incorporating the first two modules. Thus, the main objective of this work is to propose an approach that automatically assigns 100% of incoming incidents, minimizing operating costs while respecting all business requirements. As said above, this is not a theoretical problem but a real customer problem, with incidents that arise in real-time and must be assigned in a matter of seconds. Furthermore, all the results presented in this article have been implemented and are currently part of the RACC resource allocation software.

The results that have been implemented consist of an optimization algorithm that has been specifically designed to solve this problem, combining heuristics and linear programming (LP) to create an efficient and effective solution. The objective is to determine the optimal allocation of resources to incidents in real-time, minimizing a combination of service costs and arrival time, finding the right balance between cost and service according to the RACC specifications.

Although, at first glance, it might seem to be a Vehicle Routing Problem [1–3], this problem is in fact more similar to a dynamic parallel machines scheduling problem [4–6], where resources are heterogeneous machines that will be assigned to jobs as they arrive. Then, the time required for a resource to arrive to the incident's location can be understood as the set-up time of the machine.

Therefore, the main contribution of this paper is the innovative approach taken to almost optimally solve this multi-objective problem in real-time. The design of the solution and its implementation were developed for a real-world problem, which involves different characteristics and requirements to the theoretical and more limited problems.

The remainder of this paper is organized as follows. In Section 2 the relevant literature is reviewed. Section 3 is devoted to formally describing the problem scope. Section 4 presents the solution approach. Section 5 presents the computational experiments performed to calibrate the model and their results. The performance of the solution in production, and a set of benchmarks are defined. Finally, Section 6 concludes this work and proposes possible future research lines.

2. Related Literature

At first sight, the problem we face in this paper might be erroneously linked to a certain kind of combinatorial problems regarding vehicles and routes: the Vehicle Routing Problem (VRP) [1–3]. That would be true if all the problem information was well known a priori and we were interested in minimizing the sum of the vehicles route lengths; however, this is not the case. We deal with a dynamic problem, where jobs are not known until they happen, and hence we do not create an optimal route for each vehicle taking into account distances, the clients' time windows, etc.

Although there are Dynamic VRPs in the literature [7], the focus and objectives of these problems are different from ours. We react to the appearance of jobs by looking for the best assignment in that specific moment, which is measured by its contribution to the total setup time and the cost of the operations. Therefore, we consider the whole problem

closer to a scheduling one [8–10], where breakdowns (or more generally, incidents) are jobs, and the cars/trucks sent by the company to fix them or tow the vehicle to a garage are machines. Each of the steps that compose this scheduling involve an allocation of a resource to an incident.

Accordingly, the problem can be considered a parallel machine scheduling problem [4,5], where jobs are processed by one machine and the system contains several machines ready to process jobs. We have heterogeneous machines, and jobs arrive dynamically in real-time. Each job consists of a single task (i.e., an incident), and can be assigned to a specific set of machines (those resources with the necessary skills to do the repair).

The setup times and processing times are different for each job-machine pair. The setup time is the time that it takes for the resource to move from its current position to the location of the incident. The resources' skill level determines the processing time to solve the incident, which can be an on site repair or a tow to the garage. The objective function aims to minimize both the total setup time and the cost of the operations. Thus, a trade-off between setup times and costs must be found. This is the basis of our problem.

Scheduling problems are known to be NP-Complete. If all the information is given in advance, and then the optimization procedure can be run off-line, it is referred to as static scheduling. However, if part of the information is not known until some events happen, then we will be facing a dynamic scheduling problem [6]. Our problem belongs to the second type. In this case, the problem complexity is NP-Hard, since the special case in which machines are identical, and the information is known in advance is also NP-hard [11,12].

Some authors [13–17] defined and classified dynamic scheduling problems considering different features: the nature of the real-time events, the rescheduling strategies, and the rescheduling triggers. They gathered and classified solution approaches and techniques used in different papers.

In relation to the nature of the events happening in the system, there are two different kinds of events: resource-related and/or job-related events. In our case, both possibilities coexist. The later are the most common ones, i.e., the appearance of new jobs (emergencies/breakdowns). However, resources may also generate events, such as unexpected temporary cuts in driver availabilities.

Rescheduling strategies happen when a real-time event appears and the current schedule has to be redone. In the literature, two strategies are considered: a complete rescheduling and a simple repair. We do not work with a "current schedule" and modify/redo it when rescheduling is triggered, but build it in a completely reactive scheduling approach, as mentioned before. Since we may reschedule those jobs that are already allocated but did not start being processed, a repair strategy is the one that fits the best with our problem.

Another important decision to consider is when to reschedule, i.e, what rescheduling triggers are taken into account. The related literature includes three possibilities: periodic reschedules, event-driven schedules, and hybrid solutions. In our case, given that the system is not extremely busy, periodic reschedules are done every minute. This is a balance, since it allows the system to gather a few incidents to run the optimization, instead of always making greedy local choices for single jobs, and, on the other hand, does not delay the company response to customers at all.

Regarding general approaches to dynamic scheduling, Ouelhadj and Petrovic [13] proposed three possibilities: completely reactive scheduling (on line), robust pro-active scheduling, and predictive reactive scheduling. Our case is clearly the first, since no firm schedule is generated in advance and all decisions are made locally in real-time. The same paper classifies the techniques used to tackle dynamic scheduling problems into five groups: heuristics, metaheuristics, multi-agent based solutions, multi-agent scheduling architectures, and other Artificial Intelligence techniques. In this sense, we are presenting a hybrid approach in which we combine heuristics and linear programming. A deeper explanation of the problem and proposed solution can be found in the next sections.

To the best of our knowledge, no published works deal exactly with the same problem we are presenting, with the exception of a few simplified examples. Therefore, we have selected some recent ones that, despite not facing roadside assistance issues, are conceptually similar to ours. Ozbay et al. [18] applied scheduling and allocation to traffic incident management, where the authors proposed different MILP (Mathematical Integer Linear Programming) models with probabilistic constraints in order to address two sub-problems: incident response and resource allocation for traffic incident management. In the former, they solve the problem statically, taking into account the stochastic resource requirements at the sites of the potential incidents. In this sense, our approach is more accurate considering real-time incidents and solving the problem dynamically.

Other works regarding emergency response management systems are discussed in Mukhopadhyay et al. [19], where the authors drew the emergency response pipeline combining several models depending on certain criteria: static vs. dynamic work, allocation or dispatching-oriented, etc. Most referenced papers work on stages related to allocation rather than the real-time scheduling, which we are focusing on.

Apart from emergency response management, parallel machine scheduling is mainly found in two industrial fields: industrial production and computing (more specifically, operating systems). There are also some papers that directly work on the generic scheduling problem without contextualization. It is in these fields where we find closer examples to ours.

In Barbosa and Moreira [20], the authors proposed a dynamic scheduling method that repairs the predictive schedule when new jobs are submitted. The scheduling method is divided into a scheduling strategy and a scheduling algorithm based on an adaptation of the Heterogeneous Earliest-Finish-Time (HEFT) algorithm, called P-HEFT, to handle parallel tasks in heterogeneous clusters while optimizing the makespan.

Yu et al. [21] studied an agent-based scheduling problem of two identical parallel machines. In this case, different experiments are performed, where jobs can be processed by one or two machines. These jobs are ready to be processed at time 0, and hence, a predictive-reactive strategy is used to define the scheduling. In the field of operating systems, Feldmann et al. [22] presented an online tailor-made algorithm to schedule jobs on parallel machines with different topologies. One of the main differences regarding our work is that job resource requirements are known beforehand, while their running times are not. Similarly, in a more recent paper, Fu et al. [23] defined a master–slave genetic algorithm to determine the job assignments, job sequence, and resource allocation. Unlike our work, there is a dynamic resource allocation for jobs that need additional resources, which allows assignments and reassignments.

Another similar approach can be found in Wu and Che [24], where a memetic differential evolution algorithm was proposed to solve an unrelated parallel machine scheduling problem following the objective to minimize both the makespan and total energy consumption. Finally, Cheng and Huang [25] proposed a hybrid genetic algorithm with a self-adaptive releasing time control (GARTC) to find a near-optimal solution in Just-in-Time scheduling, attempting to minimize the total earliness and tardiness delivery.

As mentioned before, none of these examples tackled the same problem we present in this paper. These papers provide us with successful approaches that have already been applied to similar problems.

3. Problem Scope and Definition

The problem presented in this work consists of two key elements to handle: incidents and resources. Examples of roadside incidents could include the replacement of a tire or battery, a fuel or a towing service. These can happen at any time, any day of the year, for any location within a specific area. This case-study has been solved for the region of Catalonia, however, is easily scalable to other regions of Spain where the client currently operates.

Regarding resources, they are geolocated or are assumed to be found at a base location, and each one has a set of specific skills determining the type of roadside service it can be assigned to. Resources owned by the company are prioritized. However, there are others outsourced to external providers. Costs related to the company's resources are lower than those of the providers' fleets.

The allocation of resources to roadside incidents must be compliant with specific requirements indicated by the client that are classified into five categories, as shown in Figure 2.

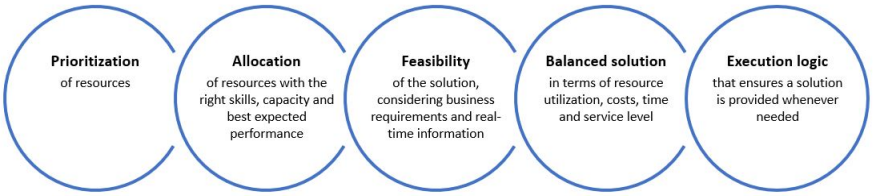

Figure 2. Main requirements.

- Ensuring the right prioritization
  - Resource priority.
    Resources are prioritized as follows (from higher to lower priority): (i) Resources owned by the client, (ii) freelance tow trucks, and (iii) other external providers.
  - Resource preferred by the client.
    If there is a resource preferred by the client, it will have higher priority than the others as long as it does not compromise the service level. If the preferred resource is available and arrives on time, no optimization is required, unless there is a conflict with some other incident requiring the same resource.
  - Resource exclusive for the client.
    There are some clients with exclusive resources assigned in advance. In this case, these will have higher priority if it does not compromise the service level.
  - Resource excluded for a client.
    There are some resources that cannot be assigned to specific clients.
- Allocating the right resource
  - Resource capacities.
    Resources owned by the client and freelance tow trucks are individual resources. This means that they can only attend one incident at a time. External providers are third parties that are (theoretically) considered to have infinite capacity.
  - Resource skills.
    For a resource to be considered a candidate for an incident, its skills must meet (at least) those required by the incident. This is the case for all resources, including external providers, which usually have a broader set of skills as they include a whole fleet of resources owned by the third party.
- Finding an optimal and feasible route
  - Services with more than one location.
    This happens when the vehicle involved in the incident needs to be moved from the original location of the incident to a repair shop. In these cases, the service distance is not only the distance between the current position of the resource and the incident's location, but also the distance between the incident's location and the repair shop.
  - Flexible work calendar and extra hours.
    Each resource has its own availability and willingness to work extra hours.
  - Arrival time at the location of the incident.

There is an upper bound that, when exceeded, leads to high financial penalties. This is tightly linked to the resources location at the moment the incident occurs and to the traffic situation.
  - Resources return to their base location at the end of each shift.
    Some time must be saved at the end of each shift so that each resource is able to arrive at its base location on time. This affects the solution as a resource that is sent to cover incidents that are far away from its base location will need more time to go back than one that covers incidents closer to the base location.
  - Real-time traffic.
    The optimal solution can be significantly different depending on the traffic status. Times and costs are computed using information retrieved from the `Google Distance Matrix API` (https://developers.google.com/maps/documentation/distance-matrix/overview accessed on 18 August 2021) in real-time.
- Providing a balanced solution
  - Balanced allocation of resources.
    Some resources have a broader set of skills or skills that are required more frequently than others. The solution must avoid over-loading some resources while others remain idle most of the time.
  - Balancing cost vs. arrival time.
    The minimization of operational costs and the compliance with the client's SLA are two objectives that do not always lead to the same optimal solution.
  - Service level.
    The company expects to ensure an overall service level above a certain threshold. This threshold can be violated, leading to an additional cost in the objective function. There is a maximum arrival time of 90 min that can never be exceeded. If exceeded, incidents will be manually assigned.
- Defining a realistic and failure-proof execution logic
  - Algorithm execution frequency. The optimization algorithm can run when a change on the status of a Work Order (WO) happens, when a new WO is created, or after a certain time period is elapsed (every minute as default configuration, except if the previous execution is not finished, then it will be when it finishes).
  - Possibility to reallocate a WO that is already planned but still not in progress. This can be due to the fact that the resource has rejected the allocation or to the fact that there is a better allocation available.
  - Possibility to allocate some WO manually. Manual allocation will override any conflicting automatic allocations.

The characteristics presented above make the resource allocation problem a key challenge to face as it directly impacts the core business of our client. To the best of our knowledge, no commercial software offers the possibility to include all these requirements in a flexible and efficient way. Henceforth, to solve this problem, a fully customized engine needs to be developed from scratch.

Formally, the problem can be stated as a multi-objective function and a set of constraints. The objectives to consider are the minimization of the operational costs and the minimization of the arrival time to incidents (time from the notification of the incident until the resource arrival to the location). The former includes resources fixed costs, resources variable costs (by km) and the financial penalty related to services that are not covered in compliance with the SLA. These costs will vary depending on the time frame (day/night shift) and on the day (business days/holidays). The arrival time can include the time until the resource is made available (if it was just about to start its shift or was working on another WO), and the travel time from its current position to the location of the incident.

4. Solution Approach

The aim of this section is to present an innovative method of resource scheduling and allocation of roadside services in real-time while minimizing the operational costs, minimizing the arrival times (which will imply an improvement of the service level), and satisfying all the requirements and business constraints.

Given that the optimization is performed in real-time, a short response time is required from the optimization algorithm. Therefore, taking into account the size of the problem and the fact that the proposed solution must be scalable to wider regions, we hypothesize that a hybrid approach, combining a heuristic approach together with an exact method, is more appropriate than an exact method. Additionally, information on the current status of the traffic is needed to perform the optimization. The `Google Distance Matrix API` is used to retrieve this information, with a significantly high response time for each candidate resource to evaluate. Hence, retrieving the time to go from each of the available resources to the incident is not an option.

Accordingly, we propose using a heuristic approach that computes a ranking of resources for each incident, prior to a linear programming optimization, which, in turn, decides the best assignment of resources to incidents. Most of the requirements are considered in the heuristic approach to generate the list of candidates for each incident: (i) the maximum arrival time is 90 min; (ii) work schedules must be complied with, with a flexibility of 15 min before and after the beginning and end of each shift, respectively; (iii) resources must be able to come back to their base location before the end of each shift; and (iv) resources have at least the same skills as the ones required by the WO for them to be a considered a candidate.

Then, linear programming ensures that: (i) the resource capacity is not exceeded (a resource can handle, at most, one incident at the same time; however, incidents can be pre-allocated to resources that will be available in the near future), (ii) the use of external providers is limited to circumstances where there are no other resources compliant with the SLA, (iii) at least one resource will be allocated to each incident, and (iv) the allocation of resources to incidents when the resources are not candidates for the incident is set to 0. The final solution was developed through multiple iterations with the client, progressively adding complexity to the logic and jointly analyzing the impact of each of the requirements into the final solution.

Algorithm 1 presents how the approach works. The procedure *SolveAllocation* expects work orders (wos) as inputs, which represent all the new incidents and all incidents that are still not accepted by resources, the list of available resources, and the following parameters: the minimum number of candidates per work order; the estimated delays for external providers; and the SLA penalization parameters. This procedure is executed every minute if the previous execution has finished; or when the previous execution finishes.

After preparing the solution structure (line 5) and obtaining the last request time (line 6), the heuristic method (lines 7 to 13) reduces the dimension of the problem allowing the use of linear programming (line 14) to solve the allocation problem. Thanks to the heuristic approach, the number of times that the `Google Distance Matrix API` is called is also reduced, which consequently dramatically shortens the execution time of the procedure, allowing a real-time response. Finally, the approach ends creating the service into the ERP system and updating the status of the resources (lines 15 and 16).

Details about the heuristic approach and the linear programming model are explained in the following.

Algorithm 1 Main Algorithm.

```
1: procedure SOLVEALLOCATION(wos, resources, parameters)
2:     wos: group of new work orders to assign
3:     resources: list of all known resources
4:     parameters: algorithm parameters (minimum number of candidates per wo, estimated delays
   for providers, and SLA penalization parameters)
5:     solution ← empty solution                    ▷ Stores the services/pairings resource-wo
6:     currentTime ← getLastRequestTime(wos)
7:     for all wo in wos do                                                    ▷ Heuristic
8:         availableResources ← getAvailableResources(currentTime, wo, resources)
9:         sortedResources ← getSortedResources(parameters, wo, availableResources)
10:        candidates ← getBestKResources(parameters, wo, sortedResources)
11:        candidatesSF ← getSFResources(parameters, wo, candidates)
12:        add candidatesSF to candidatesByWO
13:    end for
14:    bestResourcesByWO ← lpSolutionWOs(candidatesByWO, parameters)          ▷ Linear
   Programming
15:    solution ← createServices(wos, bestResourcesByWO)
16:    updateAssignedResources(bestResourcesByWO)
17:    return solution
18: end procedure
```

4.1. Heuristic

The operation of the heuristic approach (lines 7 to 13) can be summarized in four main steps.

- Step 1: For each WO, the set of available resources is computed taking into account both the availability and skills. Only resources having at least the skills required by the incident are taken into account. A resource is considered available if its shift begins in the next 15 min or has already started and not ended. Resources that are currently working on another WO are also considered. This is done in the function *getAvailableResources* (line 8).
- Step 2: Available candidates are sorted in the function *getSortedResources* (line 9). They are sorted by their euclidean distance to the incident location to create a ranking of the closest resources, which will then be further evaluated. This is done to avoid retrieving traveling time information from the `Google Distance Matrix API` for all candidates, as it takes a significant amount of time.
- Step 3: Once all available resources are sorted, the k nearest candidates using the euclidean distance and the maximum radius of x km are selected through function *getBestKResources* (line 10). The variables k and x are input parameters that the client can change at any moment, and the final number of nearest candidates is the maximum between the input parameter k and the number of incidents to be allocated. When the minimum number of k candidates is not reached within the predefined radius x, all other resources are considered and sorted by distance to select the nearest k candidates. These rankings are performed likewise and separately for each of the three kinds of resources. If there are enough candidates (k) for the first ranking, which includes the best resources owned by the company, no further rankings are computed. Otherwise, a second ranking also including freelance tow trucks is computed. If there are not enough candidates yet, the third ranking including external providers is then computed.
- Step 4: Finally, a selection factor (SF) is calculated for each resource in function *getSFResources* (line 11). Since the optimization objective is to minimize both operational costs and arrival time, the SF is calculated using both and as follows:

$$SF_{wo,r} = \frac{cost_r - \mu_c}{\sigma_c} + \frac{time_r - \mu_t}{\sigma_t} \quad \forall r \in candidates \tag{1}$$

where

- $cost_r$ is the value resulting from adding the fixed cost of the selected resource r with the product of the variable cost and the service distance (distance from the current location of the resource to the place where the incident is located and to its destination if the car needs to be moved to a second location).
- $time_r$ is the estimated arrival time for the resource r retrieved through the `Google Distance Matrix API` in real-time.
- μ_c is the mean cost value for all candidates.
- σ_c is the standard deviation for the cost of all candidates.
- μ_t is the mean time value for all candidates.
- σ_t is the standard deviation for the time of all candidates.

4.2. Linear Programming

Once the heuristic method has created the list of k candidates for each type of resource, function *lpSolutionWOs* (line 14) is run. It builds and solves the linear programming problem optimizing all the allocations at the same time. The formulation is detailed below.

Sets

- $R \in \{1..r\}$: all resources.
- $F \in R$: owned resources and freelance tow trucks.
- $O \in \{1..o\}$: group of incidents to be optimized simultaneously.
- $N = (r, o)|r \in R, o \in O$: non-considered pairs (a candidate for one WO is not necessarily a good candidate for another WO, and thus the pair would not be a valid choice).
- $P = (r, o, k)|r \in R, o \in O, k \in R$: all combinations of external provider or freelance tow truck over the SLA, incident and freelance tow truck under the SLA. P is generated by iterating over the subset of incidents that have at least one freelance tow truck under the SLA as a candidate. For each of these incidents and for each freelance tow truck that respects the SLA for the corresponding incident, a triplet is created for each external provider or tow truck over the SLA that are also candidates for this same incident.

Parameters

- c_{ij}: SF of the pair formed by a resource $i \in R$ and an incident $j \in O$.

Variables

- x_{ij}: binary decision variable; its value is 1 when the resource $i \in R$ is allocated to the incident $j \in O$.

$$\min_x \sum_{\forall i \in R} \sum_{\forall j \in O} c_{ij} \cdot x_{ij} \tag{2}$$

$$\text{s.t.} \sum_{\forall i \in R} x_{ij} = 1 \qquad \forall j \in O \tag{3}$$

$$\sum_{\forall j \in O} x_{ij} \leq 1 \qquad \forall i \in F \tag{4}$$

$$\sum_{\forall l \in O} x_{kl} \geq x_{ij} \qquad \forall (i, j, k) \in P \tag{5}$$

$$x_{ij} = 0 \qquad \forall i \in R, \forall j \in O | (i, j) \in N \tag{6}$$

$$x_{ij} \in \{0, 1\} \tag{7}$$

The objective function consists of the sum of the SFs for all pairs of resource–incident that are finally allocated. Given that the SF leverages the total cost (cost of the allocation and cost of the penalization for non-compliance with the SLA) and the arrival time to the incident location, the aim is to minimize the value of the objective function. Constraint (3) ensures that all the incidents are allocated to some resource.

Constraint (4) avoids the allocation of a finite resource (owned resource or freelance tow truck) to more than one incident at the same time. Constraint (5) sets the priority for

freelance tow trucks under the SLA over freelance tow trucks over the SLA and external providers. Constraint (6) avoids the allocation of those resources that are not candidates for a given incident. Finally, decision variables x_{ij} are defined as Boolean through constraint (7).

5. Computational Experiments

The computational experiments are divided into three blocks: Pre-Production, Production, and Benchmarks. Pre-Production includes results from the experiments that were obtained by simulating one day in the past. These one-day experiments allowed us and RACC to assess the performance of the solution proposal against real data recovered from their historical files. Once the results showed that the proposal was a good improvement, the algorithm was put into production. The results from the first two months (before COVID-19) are presented in the second block, Production. Finally, a set of benchmark problems is listed. We generated this since there were no similar benchmarks in the literature that matched our proposal.

5.1. Pre-Production

The implementation of the approach was achieved using `Python`, including the linear programming model, which requires the library `Pyomo` and the solver `glpk`. All the experiments were run locally using a computer with a Windows operating system and 32 GB of RAM. In that environment, the optimization of any allocation of incidents required around 1 s.

The approach was evaluated by simulating one day of activity and comparing the results with what happened in reality on that day. The results were thoroughly analyzed to ensure the behavior of the algorithm met all requirements even when there are candidates really close to each other in terms of objective function value.

The day of activity was selected randomly and corresponded to 17 September 2018, from 6 a.m. to 3 p.m., in the area of Catalonia (Barcelona, Girona, Lleida, Tarragona). A set of 567 incidents was extracted and used to test the algorithm, using 147 owned resources (each of which has its own working schedule), 46 tow trucks, and 427 external providers. These are the real numbers of incidents that happened and resources that were available during that day.

Figures used in this section to describe the final results are part of a user interface (UI) that was built in `R Shiny` to iterate the solution with the client through different versions of the algorithm.

Figures 3–5 show the main results in terms of resource utilization, cost by resource type and distance, before and after the optimization. The light grey shows the values from the allocation performed in reality (according to the historical data), and the dark grey shows the equivalent results obtained using the resource allocation optimization algorithm.

In terms of resource utilization:

- Work orders covered by resources owned by the company increased from 256 to 295.
- Freelance tow truck use increased in the number of incidents covered, from 96 to 168.
- The usage of external providers was dramatically reduced to more than half, from 215 incidents to 104.

We checked that our algorithm only allocated 15.7% of the incidents to the same type of resource allocated in reality. Additionally, the overall cost of the operations was reduced by 18.88% in this sample. Considering the initial cost of the client as the 100%:

- The external providers cost was reduced from 64.41% to 28.03% of the total.
- The freelance tow trucks cost was almost double since they almost doubled the number of services. This represented 17.50% and went up to 33.19% with the optimization.
- The cost of resources owned by the company increased slightly from 18.09% to 19.90%.

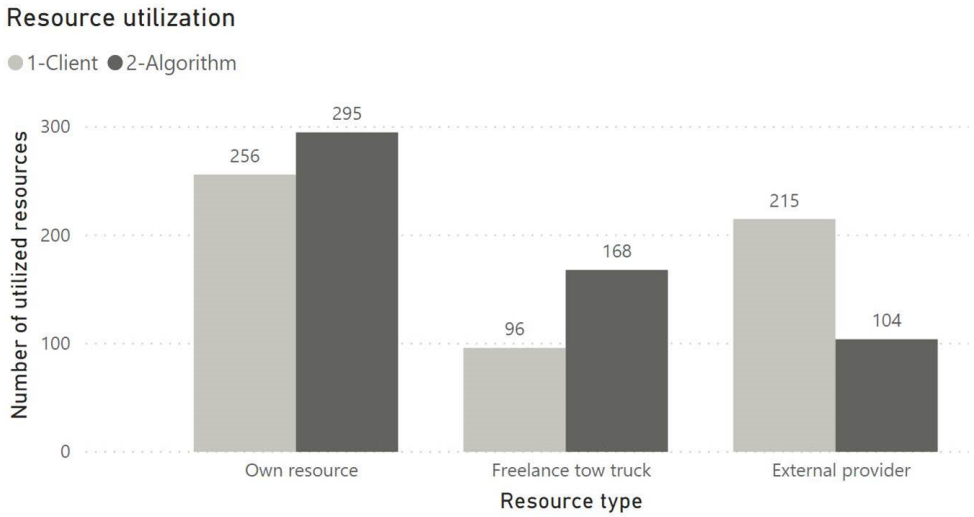

Figure 3. Resource utilization before (light grey) and after (dark grey) the optimization.

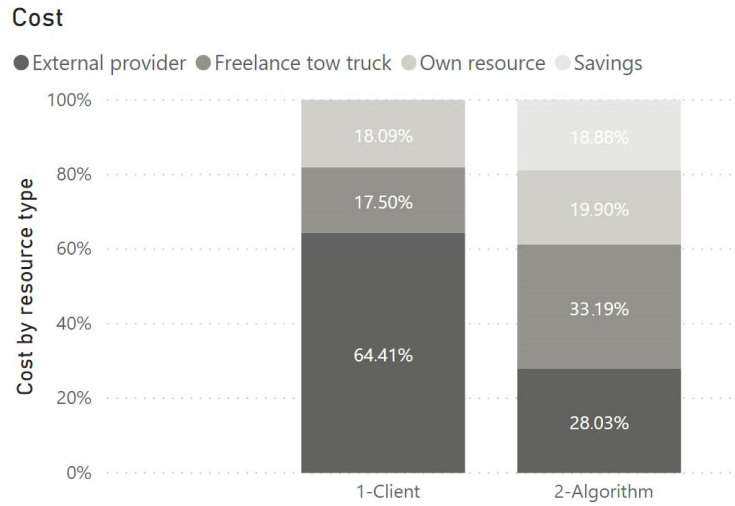

Figure 4. Total Cost savings and evolution of the percentage by resource type.

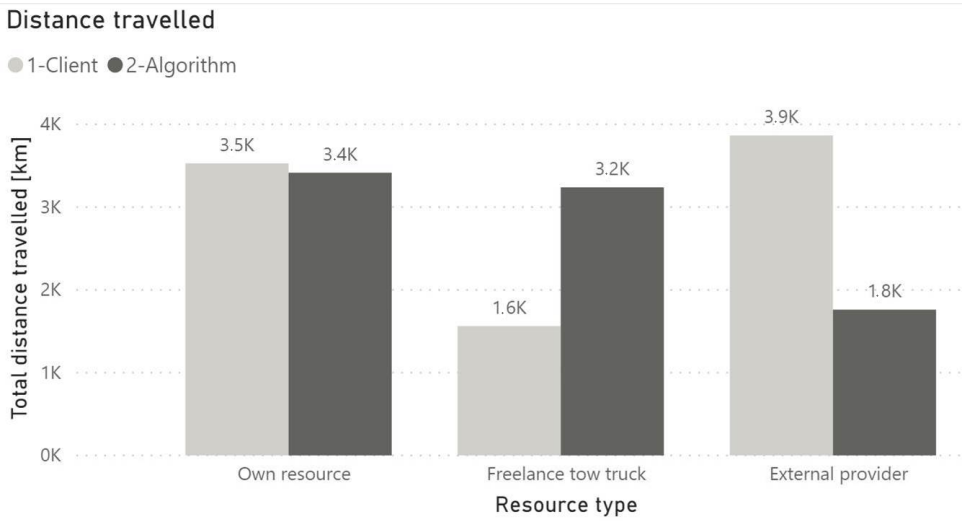

Figure 5. Distance by resource type, before (light grey) and after (dark grey) the optimization.

Finally, the overall distance traveled was reduced:

- The distance traveled for the owned resources was reduced slightly (from 3530 Km to 3416 km) although the number of services covered increased, which means that these resources increased their productivity while reducing their movements.
- The freelance tow truck distance traveled was doubled (from 1562 to 3239 km) since they almost doubled the number of services.
- The external providers distance traveled was reduced to less than half of what they were doing in reality from 3866 to 1761 km.

Figure 6 shows an example of the ranking created for one specific incident. In this ranking, the selection factor leverages the cost and time of arrival to determine one single metric that is used to sort the resources from the best (most negative) to the worse (Step 4 of the heuristic described in Section 4).

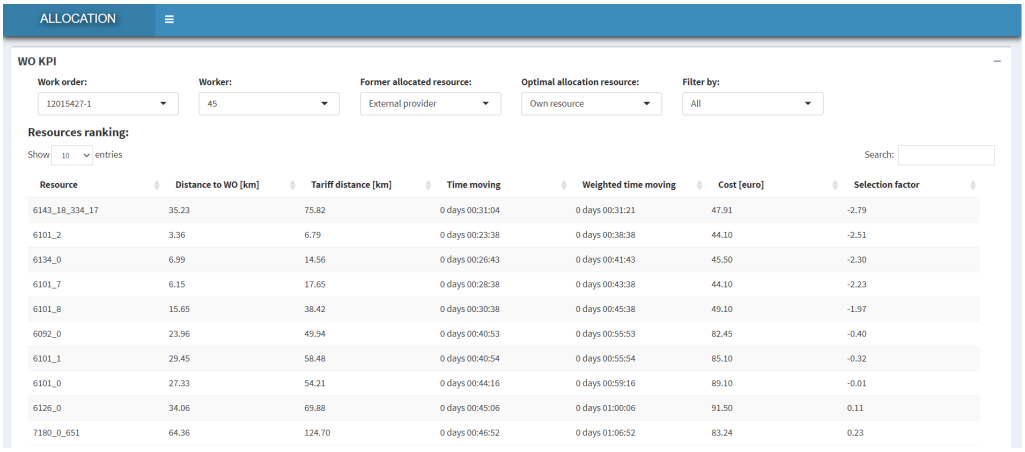

Figure 6. Ranking of the top resources for one specific incident.

5.2. Production

The shift to production was performed during the first half of 2020 in a phased manner, starting in January in the Balearic Islands. We consider January, February, and March the pre-COVID19 period, since the workload and performance were generally representative of a typical year.

The automatic allocation of mechanical incidents performed using the former system was around 40%, and a portion of these automatically allocated incidents were either allocated to resources that lacked some of the required skills or to resources that were not available for some reason. Certain times were assigned to external providers even when an owned resource could have provided the service. With this new solution, the automatic allocation was already over 80% during the first months in production, with fewer misallocations due to unexpected situations and data quality issues (incomplete WO information and partial or misspelled addresses, wrong or a lack of geolocation data, etc.). This percentage is expected to increase once the data quality issues are resolved and the algorithm is fine-tuned to handle these unprecedented situations.

It is important to highlight that there are incidents that are not linked only to one location but two, which means that the vehicle will need to be towed from the first to the second. Until now, the incident would be assigned to the freelance tow truck or external provider nearest to the first coordinate, regardless of where the second coordinate was. However, the pricing of these resources is directly proportional to the total kilometers traveled from when they leave until they return to their base, which led to very high costs for this type of service. The new algorithm takes this into account and, therefore, might select a tow truck that is between the two coordinates or even further away if the arrival time is good enough and the allocation is cheaper. The client estimates savings linked to the reduction of the operational costs for this kind of services to be around 15 to 20% on average, going up to 200% in certain cases.

To properly understand the following results, it is important to define the "coverage". This is defined as the percentage of repairable incidents that are covered by resources owned by the client out of the total number of repairable incidents received. The higher the coverage, the more efficiently resources are used. Figure 7 shows a comparison of the number of repairable incidents and the coverage that was recorded from February to March in 2019 and 2020. During these months, while February and March show a significant drop in the number of repairable incidents, the percentage of coverage went from an average of 16% to 22% across the two months, which represents an increase of 37.5% for the coverage.

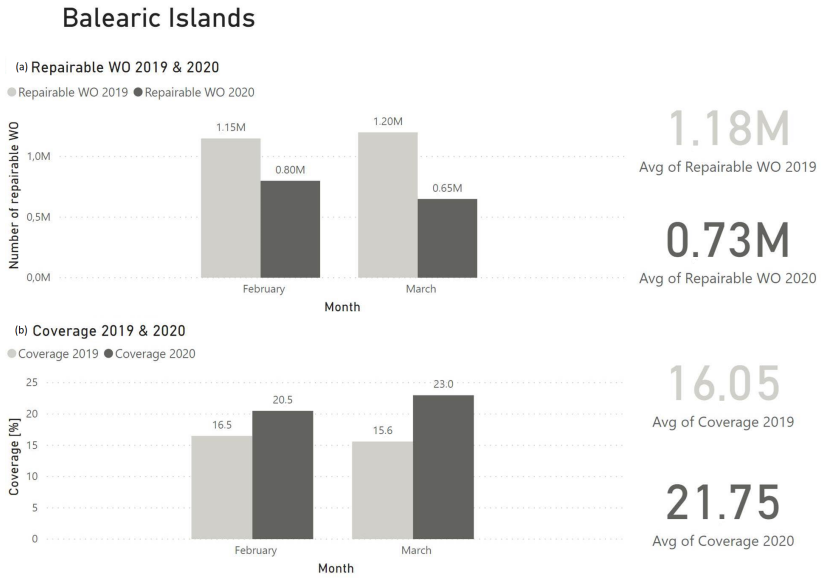

Figure 7. Number of repairable incidents (**a**) & coverage (**b**) on the Balearic Islands, 2019 vs. 2020.

5.3. Benchmark

The aim of this section is to provide a benchmark set for this problem (small, mid, and large instances) and to test our proposed approach with it. Two separate experiments are proposed: a single run of the algorithm to allocate one or more incidents, and multiple runs of the algorithm to solve, in real-time, the allocations of a certain time interval. The first is used to assess the performance of the algorithm in terms of the goodness of the solution, the value of the objective function, and the execution time. The second is used to test its capacity to run in real-time, since the allocations performed in one instant will directly affect those performed over the next minutes.

Some simplifications were made in order to make the experiments easier to reproduce. These can be classified in the following categories: pricing of the resources, penalization of freelance tow trucks and external providers, and shift flexibility.

- The pricing of resources was done through API calls to the client's pricing system. In this case, an estimate of the average fixed and variable costs for each kind of resource will be given and used to run the benchmarks.
- Freelance tow trucks and external providers were assumed to have a fixed delay of 20 and 30 min, respectively, which was added to the expected arrival time retrieved from Google. This assumption was removed to run the benchmarks.
- Shifts and breaks have, in reality, a flexibility of 15 min, and employees can work extra-hours at an extra cost. These assumptions were removed for simplicity reasons.

Additionally, the following information and parameters were needed in order to run the benchmarks and obtain comparable KPIs:

- The maximum arrival time for a resource to be valid was 90 min.
- The time required to load or unload a vehicle was 10 min. This was done every time the incident was non-repairable and there was triangulation (two coordinates

representing the origin of the incident and the destination to which the vehicle will need to be towed to).

- The penalization for going over the 30 min SLA was computed as follows:
  - 0 if the arrival time is below the SLA;
  - $min\left[40, \left(\frac{arrivaltime - SLA}{2}\right) + p_0^{\left(\frac{arrivaltime - SLA}{p_1}\right)}\right]$, otherwise.

  where $p_0 = 5$ and $p_1 = 35$ are the penalization factors, SLA is 30 min, and arrival_time is the arrival time in minutes. This function and parameters were created together with the client in order to approximate their current cost per minute over the SLA. The function parameters can be changed by the client whenever the current values no longer reflect reality.
- Owned resources do not have a fixed cost. The fixed cost for freelance tow trucks and external providers was assumed to be 25 €. The variable cost for all resources was 0.75 €/km.

5.3.1. Single-Run Experiments

Table 1 describes the inputs generated for the single-run experiments. There are three sizes (small, medium, and large), different numbers of work orders, and different numbers of available resources that are specified as owned/freelance tow trucks/external providers.

Table 1. Single-run benchmarks.

ID	Size	N° of WO	N° of Resources	Description
01	Small	3	3/3/5	Barcelona, 15 August 2018
02	Small	3	3/3/5	Outskirts of Barcelona, 15 August 2018
03	Small	3	2/1/5	Costa Brava, 15 August 2018
04	Medium	8	7/5/10	Barcelona, 4 September 2018
05	Medium	8	7/5/10	Outskirts of Barcelona, 28 August 2018
06	Medium	7	5/1/10	Costa Brava, 15 August 2018
07	Large	15	20/10/20	Barcelona, 4 September 2018
08	Large	14	15/10/20	Outskirts of Barcelona, 5 September 2018
09	Large	14	10/10/20	Barcelona, 27 August 2018

Input data for each instance includes four files:

- workorders.csv includes the incidents, and the information is described in Table 2.
- resources.csv includes the resources, and the information is described in Table 3. Notice that 862471001 is for external provider, 862471002 for tow truck, and 862471003 for owned resource.
- calendars.csv specifies the resource availability for those resources that are not available 24/7, as described in Table 4.
- current_time.csv contains a single cell indicating the time at which the optimization takes place.

Table 2. Columns in file workorders.csv.

Column Name	Description
workorder_id	ID of the WO
start_time	Time for which the service is expected
end_time	Maximum arrival time
latitude	Latitude of the incident
longitude	Longitude of the incident
destination_latitude	If specified, latitude of the destination
destination_longitude	If specified, longitude of the destination
postal_code	Postal code of the WO's location
estimated_duration	Estimated duration of the WO
skills	Skills required to solve the incident
product	ID of the type of service to be provided

Table 3. Columns in file resources.csv.

Column Name	Description
res_id	ID of the resource
res_type	Type of the resource
latitude	Current latitude of the resource
longitude	Current longitude of the resource
base_latitude	Latitude of the resource's base location
base_longitude	Longitude of the resource's base location
skills	Resource skills
is_24h	1 if the resource is available 24 h, 0 otherwise

Table 4. Columns in file calendars.csv.

Column Name	Description
res_id	ID of the resource
start_time	Shift start time
end_time	Shift end time
duration	Shift duration
time_off	If true, the resource is not available at the moment

Two outputs are provided for each instance using two files.

- benchmark_kpis.txt includes the main KPIs of the execution, as described in Table 5.
- allocated_resources.csv includes the allocation by WO, as described in Table 6.

Table 5. Columns in file benchmark_kpis.txt.

Column Name	Description
total_execution_time	Total execution time
solve_time	Solve time of the linear programming
objective_function_value	Total value of the objective function
total_cost	Total cost of the services
arrival_time	Mean arrival time
n_wo	Number of WO optimized
n_owned_resource	Number of owned resources
n_tow_truck	Number of tow trucks
n_external_provider	Number of external providers

Table 6. Columns in file allocated_resources.csv.

Column Name	Description
workorder_id	ID of the WO
execution_time	Date and time of the start of the execution
resource_id	ID of the resource
type	Type of resource
dt_displacement	Time of displacement between the current location and the incident
dt_arrival	Total time of arrival
total_cost	Total cost of the service, including SLA penalization if needed
allocation_cost	Allocation cost of the service
penalty	SLA penalty
distance	Distance from the resource's current position to the incident location
distance_tarif	Distance used to compute the allocation cost
selection_factor	Selection factor balancing time and cost
time_factor	Time factor, after standardization
cost_factor	Cost factor, after standardization
sf_offset	Offset added to the selection factor so that each resource type is prioritized
is_under_sla	True if the resource is compliant with the SLA, false otherwise
is_busy	True if the resource is busy when the allocation occurs, false otherwise
start_time	Start time of the WO
end_time	End time of the WO
is_preallocation	True if the WO is preallocated to a busy resource, false otherwise

A summary of the main results is shown in Table 7. As can be seen, the increase of execution time when the number of grouped incidents increases is not really a problem. The total execution time and solving time are quite similar because the number of resources and location impacts the execution time.

Table 7. Single-run benchmark main results.

ID	Execution Time [s]	Solve Time [s]	Objective Function Value	Total Cost	Mean Arrival Time [min]
01	2	2	−2.29	92.15	16.70
02	4	3	−0.09	117.58	22.02
03	2	2	−2.31	468.17	40.68
04	5	5	13.70	189.83	15.48
05	6	6	6.47	310.58	21.05
06	6	6	0.47	606.24	20.82
07	9	9	11.46	344.88	17.39
08	9	9	2.99	361.55	20.31
09	9	9	33.37	521.26	17.87

5.3.2. Multi-Run Benchmark

Table 8 describes the inputs generated for the multi-run experiments. Similarly to the single-run experiments, there are three sizes (small, medium, and large), different number of work orders, and different number of available resources that are specified as owned/freelance tow trucks/external providers. The files provided as input are identical, the only difference being that the time at which each work order arrives might be significantly different, which makes it necessary to run the algorithm in real-time, grouping incidents that happen approximately at the same moment.

Table 8. Multi-run benchmarks.

ID	Size	N° of WO	N° of Resources	Description
01	Small	63	5/5/30	Costa Brava, 10 September 2018, 7 a.m.–7 p.m.
02	Small	74	5/5/30	Costa Brava, 3 September 2018, 7a.m.–7p.m.
03	Small	65	5/5/30	Costa Brava, 30 July 2018, 7 a.m.–7 p.m.
04	Medium	269	10/10/30	Barcelona, 3 September 2018, 7 a.m.–7 p.m.
05	Medium	243	10/10/30	Barcelona, 10 September 2018, 7 a.m.–7 p.m.
06	Medium	238	10/10/30	Barcelona, 30 July 2018, 7 a.m.–7 p.m.
07	Large	767	35/25/100	Catalonia, 3 September 2018, 7 a.m.–5 p.m.
08	Large	641	35/25/100	Catalonia, 10 September 2018, 7 a.m.–5 p.m.
09	Large	629	35/25/100	Catalonia, 30 July 2018, 7 a.m.–5 p.m.

The outputs for the multi-run experiments include the same outputs in the single-run, with one additional column in the benchmark_kpis.txt: n_unassigned, which indicates the number of WO that could not be allocated (because of a skill that none of the available resources had). There is one additional output, occupation_rates.csv, which includes the occupation ratios of owned resources, as described in Table 9.

Table 9. Occupation_rates.csv.

Column Name	Description
res_id	ID of the resource
t_busy	Total time busy
t_available	Total time available
occupation_ratio	Occupation ratio, computed as the percentage of time available that the resource is busy
t_moving	Total time moving
t_working	Total time working
moving_ratio	Moving ratio, computed as the percentage of time available that the resource is moving
working_ratio	Working ratio, computed as the percentage of time available that the resource is working

A summary of the main results is shown in Table 10. As can be seen, the average solve time for each PO is usually very low, except for the instances with a large amount of incidents. This is due to the number of incidents in each execution of the algorithm and also on the number of available resources.

Table 10. The multi-run benchmark main results.

ID	Execution Time [s]	Solve Time/ WO [s]	Objective Function Value	Total Cost	Mean Arrival Time [min]
01	64	0.48	−131.88	3587.46	24.58
02	89	0.85	−140.39	5056.79	29.84
03	70	0.70	−80.49	2540.94	14.68
04	270	0.93	−484.40	9864.19	18.89
05	247	0.88	−433.53	6825.44	17.33
06	236	0.86	−495.86	6594.45	17.66
07	642	1.93	−1373.20	42,368.39	23.21
08	578	1.70	−1059.22	30,084.68	20.50
09	847	2.89	−1229.14	38,668.21	30.44

6. Conclusions

In this paper, we proposed an innovative approach to solve the real-time allocation of resources to incidents combining a heuristic approach with linear programming to obtain a globally optimal solution in an efficient way. The proposal leverages real-time data feeds, such as the current traffic status and each resource's position and working status to first

create a candidate list of the best potential resources. Then, it evaluates and selects the best candidate for each of the multiple concurrent incidents that are optimized simultaneously, thus, leading to the optimal global solution.

Through a one-day simulation, the performance of the algorithm was tested and compared to the real decisions made by the current system in charge of the allocation. The obtained results show the great potential of this approach: operational costs were reduced by around 19%, the use of external providers was reduced to half, and the productivity of each owned resource saw a significant increase. Then, the step into production during the months of February and March for certain regions in Spain further confirmed these results with an increase of 37.5% in coverage.

Finally, a set of benchmarks was provided in order to assess the performance of the algorithm both in single runs (the simultaneous optimization of multiple incidents) and in simulation runs (runs that simulate a real-time optimization throughout one day). Single run benchmarks showed that the increase of execution time when the number of grouped incidents increased was not a problem. Simulation runs showed that, in reality, few incidents were solved at the same time, and the average solving time for each WO was very low.

This work will be further developed with improvements regarding the cost function and the business requirements, learning from the new situations that may arise from its functioning into production. Then, the prediction of the number of incidents by region and type will be developed (the first module shown in Figure 1), as it can be used, among others, to further improve the performance of the real-time resource allocation algorithm.

Author Contributions: Conceptualization, J.d.A. and D.R.; Data curation, S.O.; Formal analysis, R.B.; Funding acquisition, D.R.; Methodology, R.B.; Supervision, J.d.A. and D.R.; Writing—original draft, R.B. and S.O.; Writing—review and editing, J.d.A. and D.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Industrial Ph.D. of Government of Catalonia 2017DI092.

Data Availability Statement: The benchmark used in this work is available at https://doi.org/10.34810/data113 accessed on 18 August 2021.

Acknowledgments: This work could not be possible without the support of both the Real Automòbil Club de Catalunya (RACC), specially the Analytics and Assistance Operations departments, and Accenture team, Supply Chain & Operations Applied Intelligence.

Conflicts of Interest: The authors declare no conflict of interest. The founders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Dantzig, G.B.; Ramser, J.H. The Truck Dispatching Problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
2. Toth, P.; Vigo, D. *The Vehicle Routing Problem*; SIAM: Philadelphia, PA, USA, 2002.
3. Ky Phuc, P.; Phuong Thao, N. Ant Colony Optimization for Multiple Pickup and Multiple Delivery Vehicle Routing Problem with Time Window and Heterogeneous Fleets. *Logistics* **2021**, *5*, 28. [CrossRef]
4. Mokotoff, E. Parallel machine scheduling problems: A survey. *Asia-Pac. J. Oper. Res.* **2001**, *18*, 193.
5. Lee, J.H.; Jang, H. Uniform Parallel Machine Scheduling with Dedicated Machines, Job Splitting and Setup Resources. *Sustainability* **2019**, *11*, 7137. [CrossRef]
6. Jackson, J.R. Simulation research on job shop production. *Nav. Res. Logist. Q.* **1957**, *4*, 287–295. [CrossRef]
7. Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* **2013**, *225*, 1–11. [CrossRef]
8. Baker, K. *Introduction to Sequencing and Scheduling*; John Wiley & Sons: Manchester, MI, USA, 1974.
9. Pinedo, M. *Scheduling*; Springer: New York, NY, USA, 2012; Volume 29.
10. Werner, F.; Burtseva, L. *Exact and Heuristic Scheduling Algorithms*; MDPI: Basel, Switzerland, 2020.
11. Lenstra, J.; Rinnooy Kan, A.; Brucker, P. Complexity of Machine Scheduling Problems. In *Studies in Integer Programming*; Annals of Discrete Mathematics; Hammer, P., Johnson, E., Korte, B., Nemhauser, G., Eds.; Elsevier: Amsterdam, The Netherlands, 1977; Volume 1, pp. 343–362.

12. Graham, R.; Lawler, E.; Lenstra, J.; Kan, A. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. In *Discrete Optimization II*; Annals of Discrete Mathematics; Hammer, P., Johnson, E., Korte, B., Eds.; Elsevier: Amsterdam, The Netherlands, 1979; Volume 5, pp. 287–326
13. Ouelhadj, D.; Petrovic, S. A survey of dynamic scheduling in manufacturing systems. *J. Sched.* **2009**, *12*, 417. [CrossRef]
14. Mohan, J.; Lanka, K.; Rao, A.N. A review of dynamic job shop scheduling techniques. *Procedia Manuf.* **2019**, *30*, 34–39. [CrossRef]
15. Bukkur, K.M.M.A.; Shukri, M.; Elmardi, O.M. A review for Dynamic Scheduling in Manufacturing. *Glob. J. Res. Eng.* **2018**, *18*, 24–37.
16. Suresh, V.; Chaudhuri, D. Dynamic scheduling—A survey of research. *Int. J. Prod. Econ.* **1993**, *32*, 53–63. [CrossRef]
17. Hu, G.J.; Fu, P.X.; Wang, M.L.; Dai, Q.Y.; Song, J.C. A Review on Dynamic Production Scheduling and Solvable Algorithms. In *Advanced Materials Research*; Trans Tech Publications Ltd.: Pfaffikon, Switzerland, 2013; Volume 622, pp. 1815–1820.
18. Ozbay, K.; Xiao, W.; Iyigun, C.; Baykal-Gursoy, M. Probabilistic programming models for response vehicle dispatching and resource allocation in traffic incident management. In *83rd Annual Meeting Compendium of Papers CD-ROM*; Citeseer: Washington, DC, USA, 2004.
19. Mukhopadhyay, A.; Pettet, G.; Vazirizade, S.; Vorobeychik, Y.; Kochenderfer, M.; Dubey, A. A Review of Emergency Incident Prediction, Resource Allocation and Dispatch Models. *arXiv* **2020**, arXiv:2006.04200.
20. Barbosa, J.G.; Moreira, B. Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters. *Parallel Comput.* **2011**, *37*, 428–438. [CrossRef]
21. Yu, F.; Wen, P.; Yi, S. A multi-agent scheduling problem for two identical parallel machines to minimize total tardiness time and makespan. *Adv. Mech. Eng.* **2018**, *10*, 1–14 [CrossRef]
22. Feldmann, A.; Sgall, J.; Teng, S.H. Dynamic scheduling on parallel machines. *Theor. Comput. Sci.* **1994**, *130*, 49–72. [CrossRef]
23. Fu, Y.; Tian, G.; Li, Z.; Wang, Z. Parallel machine scheduling with dynamic resource allocation via a master-slave genetic algorithm. *IEEJ Trans. Electr. Electron. Eng.* **2018**, *13*, 748–756. [CrossRef]
24. Wu, X.; Che, A. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega* **2019**, *82*, 155–165. [CrossRef]
25. Cheng, C.Y.; Huang, L.W. Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. *J. Manuf. Syst.* **2017**, *42*, 1–10. [CrossRef]