Nuraiman, Dian; Ozlen, Melih; Hearne, John

Article

A spatial decomposition based math-heuristic approach to the asset protection problem

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Nuraiman, Dian; Ozlen, Melih; Hearne, John (2020) : A spatial decomposition based math-heuristic approach to the asset protection problem, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 7, pp. 1-8, https://doi.org/10.1016/j.orp.2020.100141

This Version is available at:
https://hdl.handle.net/10419/246411

Operations Research Perspectives 7 (2020) 100141

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

A spatial decomposition based math-heuristic approach to the asset protection problem

Dian Nuraiman[a,b,*], Melih Ozlen[a], John Hearne[a]

[a] School of Science, RMIT University, Melbourne, Australia
[b] Department of Mathematics, UIN Sunan Gunung Djati, Bandung, Indonesia

ARTICLE INFO

Keywords:
Wildfires
Asset protection problem
Emergency response vehicles
Math-heuristic
Spatial decomposition

ABSTRACT

This paper addresses the highly critical task of planning asset protection activities during uncontrollable wildfires known in the literature as the Asset Protection Problem (APP). In the APP each asset requires a protective service to be performed by a set of emergency response vehicles within a specific time period defined by the spread of fire. We propose a new spatial decomposition based math-heuristic approach for the solution of large-scale APP's. The heuristic exploits the property that time windows are geographically correlated as fire spreads across a landscape. Thus an appropriate division of the landscape allows the problem to be decomposed into smaller more tractable sub-problems. The main challenge then is to minimise the difference between the final locations of vehicles from one division to the optimal starting locations of the next division. The performance of the proposed approach is tested on a set of benchmark instances from the literature and compared to the most recent Adaptive Large Neighborhood Search (ALNS) algorithm developed for the APP. The results show that our proposed solution approach outperforms the ALNS algorithm on all instances with comparable computation time. We also see a trend with the margin of out-performance becoming more significant as the problems become larger.

1. Introduction

Wildfires are devastating natural disasters leading to loss of life and damage to ecosystems and community assets. The catastrophic Victorian wildfires, for instance, that struck south-eastern Australia in 2009 resulted in 173 deaths and the loss of thousands of homes and other built structures [6]. During such events an Incident Management Team (IMT) is responsible for planning the deployment of resources to mitigate the hazard to community assets. Such planning requires fast and accurate decision making.

The asset protection problem (APP) is a crucial problem in emergency planning during escaped wildfires. Resources, typically comprising a mix of vehicles such as tankers, pumpers, and aerial vehicles, must be assigned to visit various community assets to undertake protection activities such as clearing debris and hosing down structures within given time windows. All these activities should occur as late as possible with sufficient time for safe departure before the advent of the fire. The capabilities of the vehicles visiting a particular asset must meet the particular requirements of that asset. For example, a 4x4 tanker might be required to access a remote asset in an area without a reticulated water supply. IMT's must deploy the available resources to serve as many assets as possible under the time constraints. Community assets comprise structures such as electricity sub-stations, bridges, communication towers, schools, historical buildings, and hospitals. Each asset has an associated value of importance to the community which determines their priority in planning.

The APP was first introduced by van der Merwe et al. [17] and formulated as a mixed integer programming model. Their model allows for a heterogeneous fleet of vehicles cooperating to maximise the total value of all assets serviced. Subsequently a dynamic rerouting model was developed which allows the incident managers to respond to disruptions to their original plans [18]. The objective in this work was to

* Corresponding author at: School of Science, RMIT University, Melbourne, Australia.
E-mail addresses: dian.nuraiman@rmit.edu.au (D. Nuraiman), melih.ozlen@rmit.edu.au (M. Ozlen), john.hearne@rmit.edu.au (J. Hearne).

https://doi.org/10.1016/j.orp.2020.100141
Received 4 September 2019; Received in revised form 10 December 2019; Accepted 15 January 2020
Available online 16 January 2020

maximise the value of assets serviced, as before, while minimising the deviation from the initial plan.

The vehicle routing problem (VRP) is basically a problem of finding a set of optimal routes for a given set of customers by assigning a fleet of vehicles to deliver goods or services to meet the customers' needs. However, in the APP vehicles might not be able to visit all assets due to time constraints and a lack of resources. Thus, the APP has similar properties to the team orienteering problem with time windows (TOPTW). The TOPTW is a problem of collecting as many rewards as possible by a team from a number of locations within their time windows.

A complexity of the APP compared with a standard TOPTW is the requirement for vehicles of differing types to synchronise their visits. An asset can only be serviced if all vehicle requirements are met within the time windows, not only in terms of the required vehicle types but also the required number of vehicles of each type. While the TOPTW with time-dependent scores (TOPTW-TDS) and mandatory visits (TOPTW-MV) are two recent variants of TOPTW introduced in [15,22], the APP is a new variant of TOPTW with synchronised visits that is not widely studied in the literature [13]. The APP has similar characteristics to task synchronisation in VRP [7]. In the VRP with synchronisation, however, all customers must be serviced while in the APP a priority is to select the most critical assets subject to the collaborative resources required for each asset. This requirement must be met by those resources able, after completing their last task, to reach an asset in the appropriate time window. Thus, the APP involves additional complexity compared with the VRP with synchronisation. Since all carriers jointly maximise the total serviced assets with a central control, the APP can be categorised in the class of centralised collaborative planning [10].

Lagrangian relaxation, column generation, branch-and-price, and branch-and-cut are exact methods that have been successfully implemented to solve various routing problems in a collaborative framework [5,8,14]. Meanwhile for metaheuristics approaches, tabu search, Ant Colony Optimization (ACO), Greedy Randomized Adaptive Search Procedure (GRASP), and evolutionary algorithms are other options [1,3,9,21]. Unfortunately, none of these approaches are readily applicable to the APP due to the complexity of some assets requiring a synchronised visit by teams of vehicles with a specific combination of capabilities within a specific time window.

Decomposition is another alternative. Various decomposition strategies have been proposed to deal with large-scale VRP's. For instance, Reimann et al. [19] developed a decomposition approach based on spatially nearby tours using a Savings-based Ant System for the VRP. A different decomposition strategy, by decomposing the problem based on vehicles, to solve the capacitated VRP with time windows can be found in [12]. Furthermore, Gansterer et al. [11] used Benders decomposition and a column generation approach to handle large-scale pickup and delivery problems.

The literature related to the APP is very limited. Adaptive Large Neighbourhood Search (ALNS) is the current best algorithm to solve the APP [20]. The results show that it is able to solve a large-scale APP consisting of 100 and 200 nodes within times suitable for operational purposes compared to the commercial integer programming solver CPLEX. In another context, a recent study conducted by Liu et al. [16] reveals that the ALNS is also the best existing approach to solve the vehicle routing problem with time windows and synchronised visits (VRPTWSyn). Their computational experiments show that the ALNS outweighs the existing approaches to 30 benchmark instances consisting of 20-80 customers provided by Bredström and Rönnqvist [4] and larger instances consisting of 100 and 200 customers given by Afifi et al. [2]. Despite this very recent effort, the search for more effective and efficient solution approaches continues. Some alternative solution approaches to the ALNS algorithm have been proposed for other collaborative routing problems [10].

Current commercial solvers such as CPLEX cannot solve large-scale APP's within times suitable for operational purposes [20]. They, however, can solve smaller problems very efficiently [17]. We utilise this capability together with the fact that time-windows in an APP are spatially correlated, to develop a new approach for solving large-scale APP's in times suitable for operational purposes. This new approach is relatively quick to implement and outperforms the ALNS [20], which is currently the best method for the APP reported in the literature.

The remainder of this paper is organized as follows. Section 2 presents the model formulation. Our solution approach is provided in Section 3. Computational experiments are detailed in Section 4 and the conclusion is presented in Section 5.

2. Model formulation

This section presents a mixed integer programming model for the asset protection problem used in previous work [20] and originally developed in [17]. The only change we have made is to tighten the constraints (6) and (7). Given a set of n assets with m initial depots and a final depot, let $N = \{1, 2, \cdots, m, m+1, \cdots, m+n, m+n+1\}$ be a set of nodes, where nodes $1, \cdots, m$ are initial depots, $m+1, \ldots, m+n$ are asset nodes, and node $m+n+1$ is a final depot. All assets are required to be serviced by a subset of available vehicles with q types of vehicles. There are P_{ik} vehicles of type k available at each initial depot i. Each asset node i has an asset value denoted by ϑ_i. Each asset i must be serviced for a duration of d_i and can only be protected if all required vehicles, denoted by $R_i = \{r_{i1}, r_{i2}, \cdots, r_{ik}, \cdots, r_{iq}\}$, can synchronise their visits. Here, r_{ik} represents the number of vehicles of type k required at node i. The service at asset i must start within a time window $[o_i, c_i]$.

As not all nodes are accessible from each node i due to temporal limitations and vehicle requirements, N_i^{k+} is defined as a set of accessible nodes by vehicles of type k going out from node i. A node j can be accessed by vehicles of type k travelling from node i if $o_i + d_i + t_{ijk} \leq c_j$ and both nodes i and j have non-zero demands for vehicles of type k, i.e., $r_{ik} > 0$ and $r_{jk} > 0$. Moreover, if node i is an initial depot, node j can be accessed by vehicles of type k if $P_{ik} > 0$ and $r_{jk} > 0$. Similarly, a set of accessible nodes N_i^{k-} for vehicles of type k coming into node i can be defined by the same way.

There are four sets of decision variables used in the mathematical model. If an asset node i is serviced, the binary variable y_i takes value 1, 0 otherwise. z_{ijk} is a binary variable that takes value 1 if there are any vehicles of type k travelling from node i to j, 0 otherwise. The number of vehicles of type k travelling from node i to j is denoted by x_{ijk}. The service at node i starts at time s_i.

First we introduce the main notation used in the mathematical model.

Sets

N	: set of nodes
D	: set of initial depot nodes, where $D = \{1, 2, ..., m\}$
C	: set of asset nodes, where $C = \{m + 1, m + 2, ..., m + n\}$
V	: set of vehicle types, where $V = \{1, 2, ..., q\}$
N_i^{k+}	: set of accessible nodes from node i by vehicles of type k
N_i^{k-}	: set of accessible nodes to node i by vehicles of type k

Parameters

ϑ_i	: asset value at node i
o_i	: opening service time at node i
c_i	: closing service time at node i
d_i	: service duration at node i
r_{ik}	: number of vehicles of type k required at node i
t_{ijk}	: travel time from node i to j by vehicles of type k
P_{ik}	: number of vehicles of type k available at depot i

Decision variables

y_i	$= \begin{cases} 1, & \text{if node} i \text{is serviced} \\ 0, & \text{otherwise} \end{cases}$
z_{ijk}	$= \begin{cases} 1, & \text{if vehicles of type } k \text{ travelling from node } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$
x_{ijk}	: number of vehicles of type k travelling from node i to j
s_i	: service starting time of node i

The objective function can be written as follows

$$\max \sum_{i \in C} \vartheta_i y_i \tag{1}$$

subject to:

$$\sum_{j \in N_i^{k+}} x_{ijk} \leq P_{ik}, \quad \forall\, i \in D, \forall\, k \in V \tag{2}$$

$$\sum_{i \in N_h^{k-}} x_{ihk} = \sum_{j \in N_h^{k+}} x_{hjk}, \quad \forall\, h \in C, \forall\, k \in V \tag{3}$$

$$\sum_{i \in N_h^{k-}} x_{ihk} = r_{hk} y_h, \quad \forall\, h \in C, \forall\, k \in V \tag{4}$$

$$t_{ijk} z_{ijk} \leq s_j, \quad \forall\, i \in D, \forall\, j \in N_i^{k+}, \forall\, k \in V \tag{5}$$

$$s_i - c_i + (d_i + t_{ijk} + c_i) z_{ijk} \leq s_j, \quad \forall\, i \in C, \forall\, j \in N_i^{k+}, \forall\, k \in V \tag{6}$$

$$x_{ijk} \leq r_{jk} z_{ijk}, \quad \forall\, i \in N, \forall\, j \in N_i^{k+}, \forall\, k \in V \tag{7}$$

$$o_i \leq s_i \leq c_i, \quad \forall\, i \in C \tag{8}$$

$$y_i \in \{0, 1\}, \quad \forall\, i \in C \tag{9}$$

$$z_{ijk} \in \{0, 1\}, \quad \forall\, i \in N, \forall\, j \in N_i^{k+}, \forall\, k \in V \tag{10}$$

The goal of (1) is to maximise the total value of all assets serviced. The first constraint (2), ensures the total number of vehicles of type k going out from the initial depots never exceed the number of vehicles of type k available at the depots. Furthermore, constraint (3) ensures that the inflow and outflow of vehicles of each type at all asset nodes are conserved. Constraint (4) determines the classification of asset h as serviced or not. The binary decision variable y_h will take value 1 if and only if the exact protection requirements defined by r_{hk} are met, otherwise the value is 0. Constraints (5) and (6) define the relationship between service start times at nodes serviced sequentially taking into account the travel times and service durations. Constraint (7) ensures that the number of vehicles of type k travelling from node i to j never exceed the required number of vehicles of type k at node j. The last constraint, constraint (8), defines the time windows for each asset.

3. Solution approach

In this work we propose a spatial decomposition based math-heuristic (SDM) approach to the solution of the mathematical model as described in the previous section. Our solution approach comprises two main steps. First, the problem is decomposed into stages spatially and then each stage is solved sequentially. Second, the last nodes (assets) serviced in each stage become the starting locations of the vehicles which served them in the next stage. Optimising a stage in isolation may compromise the ideal starting locations for the subsequent stage and hence the proximity to optimality of the whole system. Thus additional objectives are introduced in the problem for each stage to deal with this. This involves biasing the solution of a given stage towards a direction (see Fig. 1) that will mean resources are in locations suitable for a good start to the task of servicing assets in the next stage. All these subproblems can be solved quickly using a standard commercial solver to yield a near optimal deployment plan for the whole landscape of assets.

The dynamic spread of wildfire across a landscape means that the time-windows suitable for servicing an asset are geographically correlated. Thus, decomposition of the landscape for solving the problem at hand is based on the opening of the time window required at each asset. The n asset nodes are divided into λ stages. As a result, the set of asset

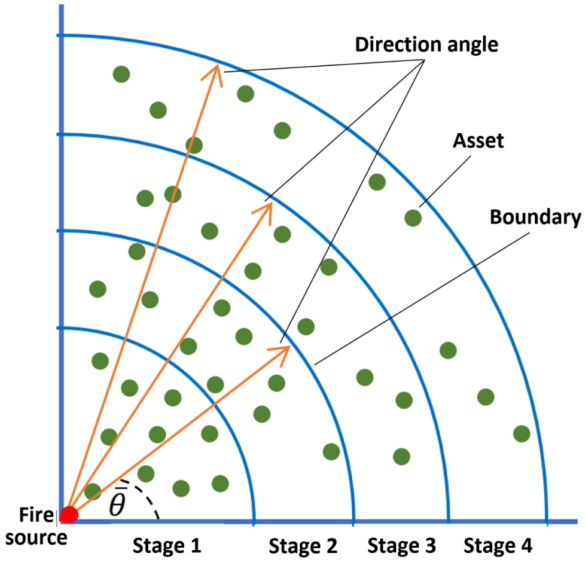


Fig. 1. Illustration of the SDM approach - a direction angle depicts the most promising direction for the next stage

nodes C is split into subsets $C_1, C_2, \cdots, C_\lambda$. Each stage has roughly the same number of nodes and λ is chosen so that this number is small enough to obtain solutions quickly for the decomposed sub-problems discussed below. In the decomposition process, the set of n nodes is sorted in ascending order based on the opening of their time windows. The first n_1 nodes are then taken as C_1 for the first stage, the next n_2 nodes as C_2 for the second stage, and so on. The problem is then solved sequentially from the first stage to the last stage using an integer programming solver like CPLEX.

In solving a stage, the role of a final depot for the whole landscape introduced at the beginning of Section 2 is substituted by a dummy final depot of the stage to fulfill constraint (3) as vehicles going out from the last serviced nodes. The dummy depot is accessible from every single node in the stage with zero travel time. In the solution process of a stage the final locations of emergency vehicles become an input for the next stage as depots. Thus, depots for each stage may be different. While the set of initial depots D is set as depots for the first stage D_1, depots for the following stage D_2 are specified based on the solution of stage 1. This information flow continues until the final stage.

For most assets the time-window during which servicing must occur is greater than the actual service time required. We use this flexibility to undertake servicing at each asset as early as possible. This allows for more flexibility in the next stage leading potentially to servicing more assets at the next stage. This is achieved by adding the secondary objective of minimising the start times of assets in stage h as follows:

$$\min \frac{\sum_{i\in C_h} s_i}{\sum_{i\in C_h} c_i} \tag{11}$$

The sum of the closing times in the denominator of Eq. (11) scales the objective to a convenient order of magnitude to achieve lexicographic optimisation of the two objectives.

Since each stage is solved independently there is potential benefit from informing earlier stages on the requirement of the following stages. To that end, we included a mechanism to nudge the final locations of the vehicles in the current stage based on the requirement of the next to improve the solution of the next stage. First, the angle of direction of each node with coordinates (x_i, y_i) in the next stage is calculated by:

$$\theta_i = \arctan\left(\frac{y_i}{x_i}\right) \tag{12}$$

A weight is then assigned to each node based on the asset value of the node and the ratio between the number of vehicles required and the number of vehicles available. This means that while node A and B have the same asset values, if node A requires fewer resources than node B, node A would receive higher priority. A target direction for stage h is then calculated by taking the weighted average of node angles in stage $h+1$ as follows:

$$\bar{\theta}_h = \frac{\sum_{i\in C_{h+1}} w_i \theta_i}{\sum_{i\in C_{h+1}} w_i} \tag{13}$$

where

$$w_i = \vartheta_i / \sum_{k\in V} \frac{r_{ik}}{\sum_{j\in D_h} P_{jk}} \tag{14}$$

This target direction that will favour the next stage is then used to influence the current stage. This is achieved by minimising the difference in angles between the departure from each node and the target direction. This strategy directs the solution as close as possible to the most promising assets in the next stage. Thus a node that is located far from the target direction will receive reduced priority for protection. A factor, $0 < \alpha < 1$, is introduced to ensure the influence of this objective is weighted considering other priorities. This step is formulated as an additional objective which is written as follows:

Input: $D, C, N_i^{k+}, N_i^{k-}, P_{ik}, \vartheta_i, o_i, c_i, r_{ik}, t_{ijk}, d_i, \alpha, \lambda$
Output: $x_{ijk}, z_{ijk}, y_i, s_i$
Sort C based on o_i in ascending order
Decompose C into λ stages
$h \leftarrow 1$
$D_1 \leftarrow D$
while $h < \lambda$ **do**
  Calculate node angles for each node in C_h by (12)
  Calculate a direction angle $\bar{\theta}_h$ by (13)
  Solve the model for stage h with the objective (16)
  Feed the solution x_{ijk} and s_i to stage $h+1$
  Assign the last serviced nodes as D_{h+1}
  Involve vehicles that are not in service and their locations into D_{h+1}
  $h \leftarrow h+1$
end
Solve the model for stage λ with the objective (1)
Integrate all solutions

Algorithm 1. SDM approach

Table 1
Data for an illustrative example.

i	x_i	y_i	d_i	ϑ_i	o_i	c_i	r_{i1}	r_{i2}
1	50	50	-	-	-	-	-	-
2	10	10	1	20	1.41421	2.41421	1	1
3	5	25	1	25	2.54951	4.54951	0	1
4	25	12	1	20	2.77308	4.77308	1	0
5	23	30	1	20	3.78021	5.78021	1	1
6	36	20	1	20	4.11825	6.11825	1	0
7	15	40	1	30	4.27200	6.27200	1	0
8	45	25	1	20	5.14782	7.14782	0	1
9	50	13	1	15	5.16624	7.16624	0	1
10	40	43	1	20	5.87282	7.87282	1	0
11	8	60	1	30	6.05310	8.05310	1	0
12	29	56	1	30	6.30635	8.30635	0	1
13	60	23	1	15	6.42573	8.42573	0	1
14	12	68	1	30	6.90507	8.90507	1	0
15	70	5	1	15	7.01783	9.01783	0	1

Table 2
Results for an illustrative example where P is the percentage of asset values protected

Objective function	Asset values	Asset values protected			
		Stage 1	Stage 2	Total	P (%)
original	310	115	65	180	58.06
improved	310	115	110	225	72.58

$$\min \sum_{i \in C_h} \alpha |\theta_i - \bar{\theta}_h| y_i \tag{15}$$

Finally, the two additional objectives, (11) and (15), are incorporated into the original objective function (1). Thus, the complete objective function for stage h can be written as follows

$$\max \left(\sum_{i \in C_h} \left(\vartheta_i - \alpha |\theta_i - \bar{\theta}_h| \right) y_i - \sum_{i \in C_h} s_i / \sum_{i \in C_h} c_i \right) \tag{16}$$

As there is no need to direct solution of the last stage and minimise the start times, this improved objective function is only used from stage 1 up to stage $\lambda - 1$.

Given a set of depots, a set of asset nodes with time windows, a set of accessible nodes from each node, the number of vehicles of each type at each depot, asset values, vehicle requirements, travel times between nodes based on travel distances, service duration for each asset, the number of stages λ, and a factor α, our solution approach is detailed in Algorithm 1. Step 9 in the algorithm means that we need to extract the final locations of vehicles from the solution of the current stage. Subsequently, the last serviced nodes and their service starting times are then fed to the next stage. Thus in step 10, the last serviced nodes become initial depots in the next stage. Furthermore, as there might be a subset of vehicles that are not in service in the current stage as a consequence of constraint (2), these vehicles and their locations must be involved in the set of initial depots in the next stage.

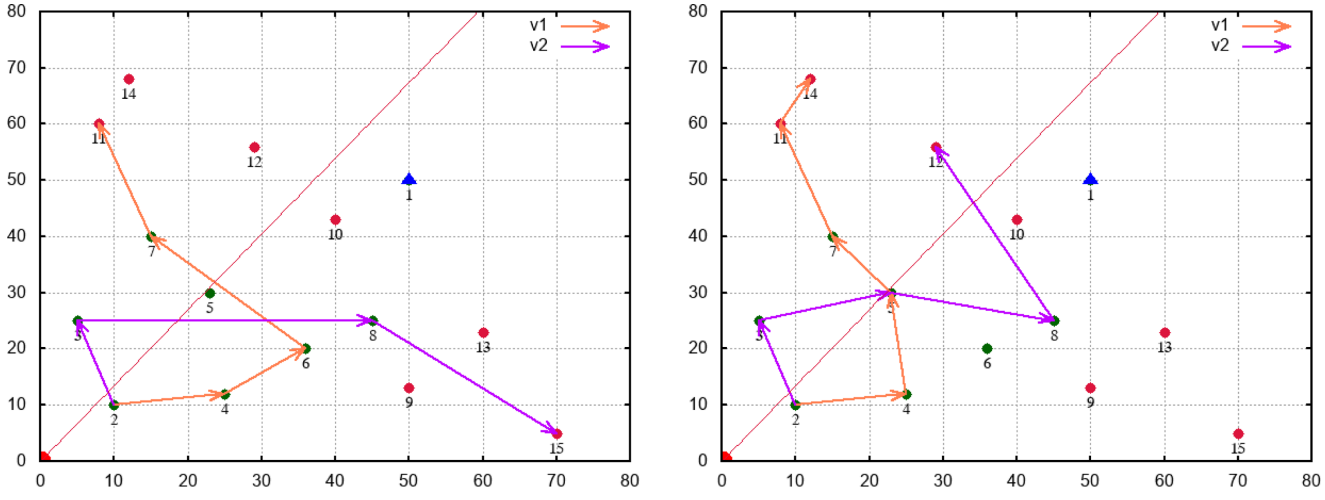
4. Computational experiments

4.1. An illustrative example

We illustrate our solution approach for a small-size instance. We consider 14 asset nodes sorted in ascending order based on the opening time windows, with node 1 as a central depot as shown in Table 1. There are only two types of vehicles available at the depot with only one vehicle of each type. The source of fire located at the origin spreads in a radial manner.

This problem is decomposed into two stages with seven nodes at each stage. To see the impact of the secondary objectives related to service starting time and directional alignment on the percentage of protected assets, we compare the use of the original objective function written in (1) with the improved objective function expressed in (16). Comparison of the results shown in Table 2 reveals that for the decomposed problem the proposed objective function can significantly improve the percentage of protected assets compared to the original objective function. The improved objective function can service 72.58% of total assets, while the original objective function can only serve 58.06% of total assets. To get better insight, solutions for both scenarios are illustrated in Fig. 2. Nodes at stage 1 are plotted in a green colour, while the red zone is stage 2. Moreover, the red line depicts the target direction based on the data in stage 2 computed by Eq. (13).

From the figure, both vehicles first service node 2 and end up at the same nodes in stage 1 for both scenarios. However, the serviced assets are slightly different. The second scenario preferred to service node 5



(a) Original (b) Improved

Fig. 2. Comparison of the performance of the two objective functions. Nine assets, including two nodes requiring a synchronous visit by two vehicles, are serviced with the new objective compared with only eight assets with the original objective.

Table 3
Comparison of the SDM performance using different values of α (C for cluster, R for random, and RC for random-cluster instances).

Instance	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
C1	59.80	60.14	60.85	61.08	61.33
C2	55.70	55.04	55.38	55.94	54.93
R1	62.52	63.46	63.35	63.45	62.50
R2	63.49	63.29	63.31	63.31	62.40
RC1	66.55	66.50	66.41	66.72	64.18
RC2	66.22	66.87	66.36	65.91	64.06
Avg.	62.28	62.55	62.61	62.73	61.57

Table 4
Comparison of the SDM performance using different values of λ (C for cluster, R for random, and RC for random-cluster instances)

Instance	λ=3		λ=4		λ=5		λ=6	
	P (%)	T (s)	P (%)	T (s)	P (%)	T (s)	P (%)	T (s)
C1	59.60	555.37	61.08	437.05	63.60	458.29	63.71	548.47
C2	56.02	394.90	55.94	321.78	57.98	267.64	58.89	187.14
R1	63.91	528.56	63.45	357.54	63.52	228.64	63.56	129.45
R2	63.36	504.32	63.31	381.08	62.33	205.24	62.75	154.69
RC1	64.41	494.23	66.72	591.11	65.84	464.99	67.10	373.47
RC2	64.97	542.53	65.91	544.10	66.10	447.45	67.86	349.85
Avg.	62.05	503.32	62.73	438.78	63.23	345.37	63.98	290.51

instead of node 6. These two nodes actually have the same asset value, but node 5 is closer to the direction angle. As a result, the vehicle of type 1 can service node 7 at an earlier service time. Furthermore, by minimising the service starting times in stage 1, the vehicle of type 1 can service not only node 11 but also node 14. Moreover, the direction angle assists the vehicle of type 2 to service node 12 instead of node 15. We can see that the improved objective function informs the first stage on what is required in the next stage.

4.2. Benchmark instances

The benchmark instances and the number of vehicles used are the same as those published in [20]. The aim is to get comparable results between solutions using the SDM approach and those obtained by ALNS algorithm as given in [20]. In general, the instances are divided into three classes: cluster, random, and a combination of random and cluster. These classes refer to the spatial distribution of data over a 140 x 140 Cartesian plane. Each class consists of two sets with 10 instances in each set. We use two types of problem size (i.e., 100 and 200 nodes) and three types of vehicles. For 100-node instances, two sets of available vehicles at a depot are used, namely (6,5,4) and (7,6,5) vehicles, while for 200-node instances more vehicles are used to tackle the increase in problem size with (9,8,7) and (12,11,10) vehicles. The numbers in parentheses refer to the number of vehicles of type 1, 2, and 3, respectively. Each problem has a single initial depot. However, due to the decomposition process, each problem will transform from a single depot problem in the first stage to multi-depot problems in the following stages. The data sets and detailed results are available at https://github.com/dn6514/Asset-Protection-Problem.

4.3. Experimental setup

The experiments were performed on a supercomputer at the Australia's high-performance computing organisation, the National Computational Infrastructure (NCI), using a single core and single thread. By design, this is the same machine and set-up as that used to obtain the ALNS results given in [20]. We use CPLEX 12.9 as the latest version available. We choose a time limit of 75 s/stage for 100 node instances and 200 s/stage for 200 node instances. These time limits are considered in order to obtain comparable computing times with the existing ALNS algorithm.

The factor α and the number of stages λ are the two main parameters of our solution approach. The first parameter plays an important role in maintaining a trade-off between the asset values serviced in the current stage and the next stage. The higher value of α, the more assets are serviced in the next stage, but fewer assets are serviced in the current stage. On the contrary, if the value of α is too small, the information of the most promising region in the next stage will be lost. As a general approach, we use a single value of α applied to all stages. We considered a number of different values for α on 200 node instances with (9,8,7) vehicles and 4 stages as shown in Table 3. The results show that the percentage of assets protected rose from $\alpha = 0.1$ and reached a peak at $\alpha = 0.4$, then dropped at $\alpha = 0.5$. Hence we choose $\alpha = 0.4$ for the next experiments.

For the second parameter, we investigated various numbers of stages on 200 node instances with (9,8,7) vehicles and $\alpha = 0.4$ as shown in Table 4. The results show that the percentage of assets serviced slightly improves as the number of stages gets larger or the size of subproblems gets smaller. The smaller the subproblem size, the faster the subproblem is solved. The SDM approach takes advantage of a constraint allowing a subset of vehicles to stay at the depot of a stage if they are not needed as written in Eq. (2), and they are available to service assets in the following stage directly.

According to Table 4, decomposing a 200-node problem into six stages with around 33 nodes at each stage is the best within the time limit given to each stage. To determine an appropriate number of stages, we choose a subproblem size of around 33 nodes. This is consistent with the experimental results by Roozbeh et al. [20] in which problems with more than 35 nodes take hours to solve. For 250-node problems, for example, we can divide them into 7 stages with around 35 nodes each stage, while for 135-node problems we can choose 4 stages with around 34 nodes each stage. For our 100-node and 200-node benchmark instances, we use $\lambda = 3$ and $\lambda = 6$, respectively, with around 33 nodes per stage.

Table 5
Results for 100 nodes with (6,5,4) vehicles (C for cluster, R for random, and RC for random-cluster instances).

Instance	CPLEX		ALNS		SDM		Improvement (%)	
	LB (%)	UB (%)	P (%)	T (s)	P (%)	T (s)	$Imp._1$	$Imp._2$
C1	26.61	90.52	60.17	138.47	65.09	95.45	144.61	8.18
C2	31.05	88.13	59.04	133.48	61.17	44.48	97.00	3.61
R1	31.79	91.36	61.19	134.47	63.68	44.30	100.31	4.07
R2	29.98	91.50	63.64	135.75	65.88	16.35	119.75	3.52
RC1	36.32	94.61	66.77	144.20	69.57	63.51	91.55	4.19
RC2	32.41	95.04	67.17	142.97	70.88	79.26	118.70	5.52
Avg.	31.36	91.86	63.00	138.22	66.05	57.22	110.60	4.84

$Imp._1$: SDM vs LB CPLEX (150s), $Imp._2$: SDM vs ALNS

Table 6
Results for 100 nodes with (7,6,5) vehicles (C for cluster, R for random, and RC for random-cluster instances).

Instance	CPLEX		ALNS		SDM		Improvement (%)	
	LB (%)	UB (%)	P (%)	T (s)	P (%)	T (s)	$Imp._1$	$Imp._2$
C1	31.11	94.45	66.66	150.39	71.34	74.98	129.32	7.02
C2	33.97	93.37	64.87	143.92	68.23	43.03	100.85	5.18
R1	32.60	96.10	68.45	138.97	71.85	26.70	120.40	4.97
R2	34.64	96.26	69.76	144.65	73.61	21.09	112.50	5.52
RC1	37.78	97.71	73.21	149.53	77.27	49.90	104.53	5.55
RC2	35.74	98.15	73.12	146.90	79.02	48.13	120.85	8.07
Avg.	34.31	96.01	69.35	145.73	73.55	43.97	114.36	6.07

$Imp._1$: SDM vs LB CPLEX (150s), $Imp._2$: SDM vs ALNS

Table 7
Results for 200 nodes with (9,8,7) vehicles (C for cluster, R for random, and RC for random-cluster instances)

Instance	CPLEX		ALNS		SDM		Improvement (%)	
	LB (%)	UB (%)	P (%)	T (s)	P (%)	T (s)	$Imp._1$	$Imp._2$
C1	20.20	100	56.13	589.60	63.71	548.47	215.40	13.50
C2	18.73	100	51.06	542.64	58.89	187.14	214.42	15.33
R1	23.18	100	58.23	539.19	63.56	129.45	174.20	9.15
R2	13.52	100	57.75	542.78	62.75	154.69	364.13	8.66
RC1	15.80	100	60.90	561.80	67.10	373.47	324.68	10.18
RC2	12.37	100	61.45	570.06	67.86	349.85	448.59	10.43
Avg.	17.30	100	57.59	557.68	63.98	290.51	269.82	11.10

$Imp._1$: SDM vs LB CPLEX (600s), $Imp._2$: SDM vs ALNS

Table 8
Results for 200 nodes with (12,11,10) vehicles (C for cluster, R for random, and RC for random-cluster instances).

Instance	CPLEX		ALNS		SDM		Improvement (%)	
	LB (%)	UB (%)	P (%)	T (s)	P (%)	T (s)	$Imp._1$	$Imp._2$
C1	26.71	100	65.11	619.33	73.51	86.40	175.22	12.90
C2	16.47	100	60.34	566.36	68.78	79.50	317.61	13.99
R1	20.98	100	69.04	585.49	77.27	59.92	268.30	11.92
R2	27.81	100	68.74	589.75	76.23	77.79	174.11	10.90
RC1	24.70	100	71.21	607.04	80.30	258.62	225.10	12.77
RC2	20.21	100	72.14	633.17	80.24	175.35	297.03	11.23
Avg.	22.81	100	67.77	600.19	76.05	122.93	233.38	12.24

$Imp._1$: SDM vs LB CPLEX (600s), $Imp._2$: SDM vs ALNS

4.4. Numerical results

In this section we first evaluate the quality of solutions produced by our solution approach for larger-size instances compared to the lower bound (LB) obtained by CPLEX in solving the model formulation described in Section 2 in a single run with comparable runtimes. The LB represents the best solution achieved under a given time limit. An improved upper bound (UB) was achieved after running CPLEX with a 10-hour time limit. We then compare the SDM approach with the current best algorithm developed for the APP, i.e., the ALNS algorithm, provided in [20]. We measure the performance of the SDM approach by calculating the percentage improvement over CPLEX single runs and the ALNS algorithm. Each table provides information on the LB produced by CPLEX with time limits of 150 s and 600 s for 100 and 200 node instances, respectively. Also shown, is the UB obtained by CPLEX with a time limit of 10 h, the percentage of protected asset values denoted by P and the computation time T in second obtained by the ALNS algorithm and the SDM approach, and the percentage improvement between them.

In general, for 100-node instances Tables 5 and 6 show an increase in serviced assets of 11.37% from 66.05% for (6,5,4) vehicles to 73.55% for (7,6,5) vehicles using the SDM approach. This represents an improvement of more than 100% over the CPLEX single runs in both cases. In constrast with the SDM, the ALNS results only improve by 10% from 63% to 69.35% of total asset values. This trend continues with an improvement of 4.84% increasing to 6.07% with an increase in the number of vehicles.

In an attempt to establish the absolute quality of our solutions we ran CPLEX with a 10-hour time limit. However, this only led to UB's of 91.86% for (6,5,4) vehicles and 96.01% for (7,6,5) vehicles.

For 200-node instances, Tables 7 and 8 show that the percentage of asset values serviced by the SDM approach improves by 18.88% from nearly 64% for (9,8,7) vehicles to just over 76% for (12,11,10) vehicles, while the ALNS obtains a lower increase of 17.67% from 57.59% to 67.77%. Compared with the ALNS, the SDM approach show an 11% improvement for (9,8,7) vehicles and a 12.24% improvement for (12,11,10) vehicles. This suggests that the SDM approach is better suited to handling a large number of vehicles than the ALNS. Our solution approach outperforms the 10-minute CPLEX solution with a significant improvement by more than 200% in both cases for different sets of vehicles. This demonstrates the capability of the SDM over previous approaches for solving large-scale problems within operational time.

Another significant finding is that the percentage improvement produced by the SDM approach over the ALNS increases as the problems get larger. The results in Tables 5 and 7 show that the SDM

approach obtains the percentage of assets protected in a range of 64%-66%. In this range of accuracy, the improvement increases by around 6% from 4.84% for 100 nodes to 11.10% for 200 nodes. For the second set of vehicles as shown in Tables 6 and 8, the SDM approach achieves greater accuracy due to the increase in the number of vehicles in a range of 73%-76%. The improvement grows by around 6% from 6.07% for 100 nodes to 12.24% for 200 nodes.

5. Conclusion

In the original work on the asset protection problem a standard application of a commercial solver (CPLEX) was used [17]. While this had planning and learning value, the solution times were too slow for practical purposes. An ALNS algorithm was later developed by Roozbeh et al. [20] which gave solutions in times useful for operational purposes. In an attempt to find a more efficient solution method we introduced the SDM approach using a commercial solver. The results with this approach consistently outperformed the published ALNS algorithm. Furthermore, the improvement produced by the SDM approach over the ALNS increases in line with the increase in the number of resources and the problem size. It is expected that the improvement will continue to grow for larger-scale problems. The approach presented in this work is applicable to similar problems where the time-windows are correlated with a direction. This is the case for evacuation problems during certain flood events, for example, or the advance of an attacking armed force. Another advantage of the SDM approach over the ALNS relates to the coding effort. New problems or the addition of more constraints are relatively easy to implement. Thus, the scalability of our method together with the use of a standard commercial solver means that new situations can be modelled quickly and solved efficiently within times suitable for operational purposes.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Dian Nuraiman: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Visualization. **Melih Ozlen:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision. **John Hearne:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition.

Acknowledgement

The authors gratefully acknowledge the constructive feedback they received from three anonymous reviewers on an earlier version of this paper which has led to significant improvements. DN is supported by the MoRA 5000 Doctors Program. JH acknowledges support for this work by the Marie Skodowska-Curie RISE H2020 project GEO-SAFE (691161).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.orp.2020.100141

References

[1] Adenso-Díaz B, Lozano S, Garcia-Carbajal S, Smith-Miles K. Assessing partnership savings in horizontal cooperation by planning linked deliveries. Transp Res Part A Policy Pract 2014;66:268–79. https://doi.org/10.1016/j.tra.2014.05.013.
[2] Afifi S, Dang D-C, Moukrim A. Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. Optim Lett 2016;10:511–25. https://doi.org/10.1007/s11590-015-0878-3.
[3] Bailey E, Unnikrishnan A, Lin D-Y. Models for minimizing backhaul costs through freight collaboration. Transp Res Rec 2011;2224:51–60. https://doi.org/10.3141/2224-07.
[4] Bredström D, Rönnqvist M. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. Eur J Operat Res 2008;191:19–31. https://doi.org/10.1016/j.ejor.2007.07.033.
[5] Chen H. Combinatorial clock-proxy exchange for carrier collaboration in less than truck load transportation. Transp Res Part E: Logist Transp Rev 2016;91:152–72. https://doi.org/10.1016/j.tre.2016.04.008.
[6] Cruz M, Sullivan A, Gould J, Sims N, Bannister A, Hollis J, Hurley R. Anatomy of a catastrophic wildfire: the Black Saturday Kilmore East fire in Victoria, Australia. For Ecol Manag 2012;284:269–85. https://doi.org/10.1016/j.foreco.2012.02.035.
[7] Drexl M. Synchronization in vehicle routing – a survey of VRPs with multiple synchronization constraints. Transp Sci 2012;46:297–316. https://doi.org/10.1287/trsc.1110.0400.
[8] Fernández E, Fontana D, Speranza M. On the collaboration uncapacitated arc routing problem. Comput Operat Res 2016;67:120–31. https://doi.org/10.1016/j.cor.2015.10.001.
[9] Gansterer M, Hartl R. Centralized bundle generation in auction-based collaborative transportation. OR Spectrum 2018;40:613–35. https://doi.org/10.1007/s00291-018-0516-4.
[10] Gansterer M, Hartl R. Collaborative vehicle routing: a survey. Eur J Operat Res 2018;268:1–12. https://doi.org/10.1016/j.ejor.2017.10.023.
[11] Gansterer M, Hartl R, Salzmann P. Exact solutions for the collaborative pickup and delivery problem. Cent Eur J Operat Res 2018;26:357–71. https://doi.org/10.1007/s10100-017-0503-x.
[12] Gulic M, Lucanin D, Skorin-Kapov N. A two-phase vehicle based decomposition algorithm for large-scale capacitated vehicle routing with time windows. Proceedings of the 35th International Convention MIPRO. 2012. p. 1104–8.
[13] Gunawan A, Lau HC, Vansteenwegen P. Orienteering problem: A survey of recent variants, solution approaches and applications. Eur J Operat Res 2016;255:315–32. https://doi.org/10.1016/j.ejor.2016.04.059.
[14] Kuyzu G. Lane covering with partner bounds in collaborative truckload transportation procurement. Comput Operat Res 2017;77:32–43. https://doi.org/10.1016/j.cor.2016.07.018.
[15] Lin S-W, Yu VF. Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. Comput Ind Eng 2017;114:195–205. https://doi.org/10.1016/j.cie.2017.10.020.
[16] Liu R, Tao Y, Xie X. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. Comput Operat Res 2019;101:250–62. https://doi.org/10.1016/j.cor.2018.08.002.
[17] van der Merwe M, Minas JP, Ozlen M, Hearne JW. A mixed integer programming approach for asset protection during escaped wildfires. Can J For Res 2015;45:444–51. https://doi.org/10.1139/cjfr-2014-0239.
[18] van der Merwe M, Ozlen M, Hearne JW, Minas JP. Dynamic rerouting of vehicles during cooperative wildfire response operations. Ann Operat Res 2017;254:467–80. https://doi.org/10.1007/s10479-017-2473-8.
[19] Reimann M, Doerner K, Hartl R. D-ants: savings based ants divide and conquer the vehicle routing problem. Comput Operat Res 2004;31:563–91. https://doi.org/10.1016/S0305-0548(03)00014-5.
[20] Roozbeh I, Ozlen M, Hearne J. An adaptive large neighbourhood search for asset protection during escaped wildfires. Comput Operat Res 2018;97:125–34. https://doi.org/10.1016/j.cor.2018.05.002.
[21] Sprenger R, Mönch L. A methodology to solve large-scale cooperative transportation planning problems. Eur J Operat Res 2012;223:626–36. https://doi.org/10.1016/j.ejor.2012.07.021.
[22] Yu VF, Jewpanya P, Lin S-W, Redi AP. Team orienteering problem with time windows and time-dependent scores. Comput Ind Eng 2019;127:213–24. https://doi.org/10.1016/j.cie.2018.11.044.