A continuous location and maintenance routing problem for offshore wind farms: mathematical models and hybrid methods

Chandra Ade Irawan[a,b,*], Said Salhi[c], Hing Kai Chan[a,b]

[a] *Nottingham University Business School, University of Nottingham Ningbo, China*
[b] *Nottingham Ningbo China Beacons of Excellence Research and Innovation Institute*
[c] *Kent Business School, University of Kent, Canterbury, UK*

Abstract

In this study, we examine a challenging green logistical problem encountered with offshore wind farms: the integrated continuous location and maintenance routing problem wherein a service operation vessel and a safe transfer boat are used to maintain offshore turbines. Our aim is to dynamically and simultaneously determine the best locations for the service operation vessel in the plane (i.e. the sea) and the best delivery and pick-up routes by which the safe transfer boat can access the turbines. An optimisation model of the problem is first developed based on mixed-integer nonlinear programming to minimise the total maintenance cost. Given the limitations of this method, a novel algorithm that integrates a genetic algorithm, variable neighbourhood search, and a Weiszfeld-based algorithm is presented. To assess the performance of the proposed technique, a guided multi-start approach and a hybrid technique based on particle swarm optimisation are introduced. Moreover, our proposed method is shown to be easily adaptable to produce results that compete with those of state-of-the-art methods when it comes to solving a related continuous location routing problem. The computational results demonstrate the effectiveness and robustness of the proposed hybrid method.

Keywords: combinatorial optimisation, continuous location, location routing, hybridisation search, offshore wind farm

1. Introduction

The demand for renewable energy is continuously increasing, and offshore wind farms are one of the most environmentally friendly sources of said energy. Thus, it is essential to enhance wind farm efficiency, making them more competitive and attractive in terms of investment and sustainability. The costs related to the operations and maintenance (O&M) of offshore wind farms, which are relatively high, are expected to account for a quarter of the life-cycle cost of each wind farm (Snyder and Kaiser, 2009). These high O&M costs are mainly due to the long turbine downtime caused by accessibility restrictions, whereby vessels are not allowed to transfer technicians during

*Corresponding author
Email address: chandra.irawan@nottingham.edu.cn (Chandra Ade Irawan)

bad weather conditions (Dewan and Asgarpour, 2016). The concept of a service operations vessel (SOV) has significant potential to reduce O&M costs by decreasing the accessibility restrictions and the transfer time, as a SOV can remain at sea for several days without needing to return to the shore base. As an example, Siemens started using a SOV (shown in Figure 1) in 2016 to maintain the turbines at the Gemini wind farm located in the northern Netherlands.

Figure 1: The SOV used in Gemini wind park (www.geminiwindpark.nl)

The SOV carries a safe transfer boat (STB) that can be used to transfer technicians and equipment to the turbines. The SOV has accommodations for technicians and can store the equipment and spare parts required to maintain the turbines. This paper focuses on minor preventive maintenance activities for which technicians and equipment are transferred by a STB that consumes less fuel and is faster than a SOV. The aim is to dynamically identify the best locations for the SOV in the plane and to generate the best routes by which the STB can transfer the technicians to the turbines. This problem can be considered as a continuous location and maintenance routing problem for offshore wind farms (CLMRPOWF), which, to the best of our knowledge, has not been investigated in the literature.

The CLMRPOWF is related to the location-routing problem (LRP) and the maintenance routing problem. For more details on the taxonomy of LRP, see Lopes et al. (2013). The LRP integrates two NP-hard problems: the location problem and the routing problem. It was initially shown by Salhi and Rand (1989) (and has since been strongly supported by many other researchers) that it can be misleading to ignore the effect of the routes when locating facilities. Several variants of LRP exist; some focus on location (e.g. type of location, size) and others on routing (e.g. classical routing, routing with pick-ups and delivery, routing with heterogeneous fleet). Some recent interesting applications of LRP are provided by, among others, Rabbani et al. (2019), Almouhanna et al. (2020), Karakostas et al. (2020), Cao et al. (2021), and Calık et al. (2021). The reader will find the frequently cited initial review by Nagy and Salhi (2007), and two interesting later reviews by Prodhon and Prins (2014) and Schneider and Drexl (2017) (with the references therein) to be informative.

To our knowledge, research on the LRP in the plane is limited compared to that on its counter-

part, the discrete LRP, in which the selected facilities are part of the potential sites. This lack of interest is mainly due to the large number of applications that consider discrete locations instead of continuous ones. Despite this obvious weakness, it is worth stressing that the solution to the planar (continuous) location problem could be used as a benchmark or an ideal solution for the discrete problem while also providing guidance when generating promising potential sites, thereby reducing unnecessary data collection and thus cost.

Schwardt and Dethloff (2005), Schwardt and Fischer (2009), and Salhi and Nagy (2009) are among the first who investigated this problem. Schwardt and Dethloff (2005) and Schwardt and Fischer (2009) proposed algorithms based on neural networks to solve the planar single-facility LRP in which the interaction between the facility and the customers is defined using interconnected neuron rings. Salhi and Nagy (2009) considered the end-points of the routes as demand points to determine the location of the facility using an algorithm known as the end-point algorithm (EPA). The new location is then fed back into the routing as the new depot location, and this process continues until there is no change in either the end-points of the routes or the depot location. Manzour-al-Ajdad et al. (2012) proposed a hierarchical heuristic-based method incorporating the EPA introduced by Salhi and Nagy (2009) to tackle the planar single-facility LRP. Ghaffarinasab et al. (2018) dealt with a planar hub LRP for the design of a parcel delivery network that is modelled using the continuous approximation technique. The problem was solved using two solution methods: an iterative Weiszfeld-type algorithm and particle swarm optimisation (PSO). Rybičková et al. (2019) designed an algorithm based on the genetic algorithm (GA) that considers the presence of multiple depots to solve the continuous LRP. Recently, Tayebi Araghi et al. (2021) investigated a green stochastic open LRP with planar facility locations. As the rectilinear distance was applied between customers' locations and the depots, a linear programming model was then adopted. For large-scale problems, a hybrid meta-heuristic method was developed instead. It is also interesting to note that our problem has some similarities to recent developments in drone routing problems, in particular the mothership and drone routing problem (Poikonen and Golden, 2020b,a; Roberti and Ruthmair, 2021), where the routing of a two-vehicle tandem is considered (for example, the truck and the drone). However, the proposed problem (the CLMRPOWF) differs from the above, as it consists of multiple separate delivery and pick-up trips, and the timing of each route needs to be determined by taking into account maintenance duration and travel time.

The maintenance routing problems faced by offshore wind farms have been investigated in the literature. However, researchers have only considered the use of crew transfer vessel (CTV) wherein the CTV navigates from a static O&M-base located at the port nearest to the wind farm. In this situation, the CTV must return to the O&M-base on the same day after the maintenance tasks are completed. As the construction of new wind farms usually takes place farther from shore, the accessibility restrictions are increased and the transfer time is longer when the above concept is applied. Dai et al. (2015) were among the first to study this problem and developed a mixed-integer linear programming (MILP) model. Stålhane et al. (2015) and Irawan et al. (2017) extended the work by

developing a Dantzig-Wolfe decomposition method (Dantzig and Wolfe, 1960). Schrotenboer et al. (2019) further explored this by developing an exact algorithm based on a branch-and-price-and-cut algorithm. Raknes et al. (2017) proposed a methodology for scheduling maintenance activities and generating CTV routes, while Schrotenboer et al. (2018) studied the routing problem by considering whether technicians should divide their time between multiple wind farms. The uncertainty related to weather conditions affects maintenance activities at offshore wind farms, and the maintenance routing problem can be enhanced by taking this uncertainty into account, as discussed by Stock-Williams and Swamy (2019), Irawan et al. (2021), and Schrotenboer et al. (2020).

We first propose an optimisation model based on a mixed-integer nonlinear programming (MINLP) model. As the model is nonlinear and difficult to solve optimally, a hybrid algorithm (HGA) that integrates a GA and variable neighborhood search (VNS) is designed. We also incorporate a new iterative approach based on the Weiszfeld algorithm to find the optimal SOV locations on the plane. This is a modification of the end-point algorithm originally developed by Salhi (1987) and applied by Salhi and Nagy (2009). We refer to this iterative approach as the modified end-point algorithm (MEPA). The performance of the proposed hybrid method (HGA) in solving the continuous problem is assessed against a guided multi-start (MS) and an implementation of PSO that we developed, which we refer to those methods as HMS and HPSO, respectively. We also design the discrete version of the problem for benchmarking purposes, which can be modelled as a MILP and solved using a commercial optimiser. Note that the proposed solution methods (HGA, HMS, and HPSO) are mainly used to solve the continuous non-linear problem. However, the proposed HGA can be used to solve the discrete problem, where its results are compared with those produced by the exact method to assess the HGA performance. In addition, to evaluate the robustness of our algorithm, our proposed HGA method is also adapted accordingly to solve a related continuous LRP, namely, the planar single-facility LRP. Competitive results are discovered when compared against those methods available in the literature. We use Thanet offshore wind farm layout to carry out computational experiments. The contributions of the study are fourfold:

- Produce a new optimisation model based on MINLP for the continuous location and maintenance routing problem
- Design a hybrid algorithm that integrates GA and VNS while introducing a novel class-based parent selection operator in the GA to increase the diversity of the population
- Develop an efficient, new, iterative approach, based on Weiszfeld algorithm to dynamically identify the best SOV locations in the plane based on a given set of routes.
- Present managerial insights obtained from extensive experimental results.

The remainder of the paper is organised as follows: Section 2 describes the CLMRPOWF and the mathematical model. Section 3 presents a description of the proposed GA, including VNS and the iterative approach for locating the SOV on the sea. In Section 4, a set of computational experiments is presented alongside the results of the MS method and the PSO algorithm. The final section summarises our findings and presents some potential research areas.

2. Problem Formulation

2.1. Overview of the proposed problem

The CLMRPOWF consists of a set of selected offshore turbines (I) that need to be maintained according to a specific schedule. The location of the turbines is fixed, and the Euclidean distance is considered. Each turbine requires an estimated maintenance duration (hours), the number of technicians needed, and the weight of the required spare parts. The technicians are usually dropped off in the morning by the STB and picked up in the evening once the maintenance activities are completed.

The SOV may travel through a number of locations that need to be determined; once the SOV is located at a certain position, the STB is then deployed to transfer crews to the selected turbines. The locations of the start and the end of the STB route may not be the same since the SOV's location may change. In this paper, the SOV consists of only one STB; however, the STB may make several trips. There are two types of trip: a delivery (drop) trip and a pick-up trip. In the former, the STB makes multiple trips - one to each turbine - to drop the technicians, usually in the morning. The latter refers to the STB's trip to collect the crews from the turbines once the maintenance activities have been completed, usually in the evening. The problem can be considered as a variant of the multi-trip separate pick-up and delivery problem with time windows (Bettinelli et al., 2019). The number of each type of trip can also be determined by the decision-maker based on the number of technicians available on the SOV.

A STB's capacity for transferring technicians and carrying equipment and parts is limited. The time it takes for the crews to board and the equipment to be transferred from the SOV to the STB is considered along with the boarding/transfer time from the STB to a turbine. The travel cost and travel time for both the SOV and STB are considered to ensure that the delivery and pick-up trips are within the given time window $[0,t_{\max}]$. A penalty cost occurs if the STB returns from the final pick-up trip later than $t_{\max}$, meaning that soft time window is implemented. Here, the time of the final pick-up trip indicates the time by which all maintenance activities have been completed. A time window is not required for delivery trip, as all such trips are performed before any pick-up trips. We also consider time synchronous between the delivery and pick-up trips as the pick-up route can be performed once the maintenance task has been completed. The pick-up time needs to consider the delivery time and the maintenance duration. The SOV cost consists of the travel cost (this is based on the fuel cost and traveling distance) and the fixed cost (this is incurred when the SOV travels from one location to another). The objective of the problem is to minimise the total maintenance cost by determining (i) the best locations for the SOV, (ii) the best routes for the STB to take to perform the maintenance activities at the offshore wind farms, and (iii) the best times for the STB to leave from and return to the SOV, including the timing when the STB needs to visit the turbines.

Figure 2 illustrates two feasible solutions for the problem in a situation where 12 turbines need to be maintained. In the solution, the SOV can be situated at several locations, with P_o and P_f

representing the start and end location of the SOV, respectively. The solution presented in Figure 2a will locate the SOV at seven sites, whereas the one in Figure 2b will locate the SOV at two sites. The start and the end points of the STB route are not necessarily the same which is illustrated in the first STB delivery trip in Figure 2a (DT1). The delivery and pick-up routes are also not necessarily the same, as maintenance duration needs to be considered. If the SOV is located in a turbine, the technicians and equipment can access the turbine via the SOV. It is assumed that the boarding and the transfer time are the same as those for the STB. In this case, the travel time is negligible and hence set to zero.

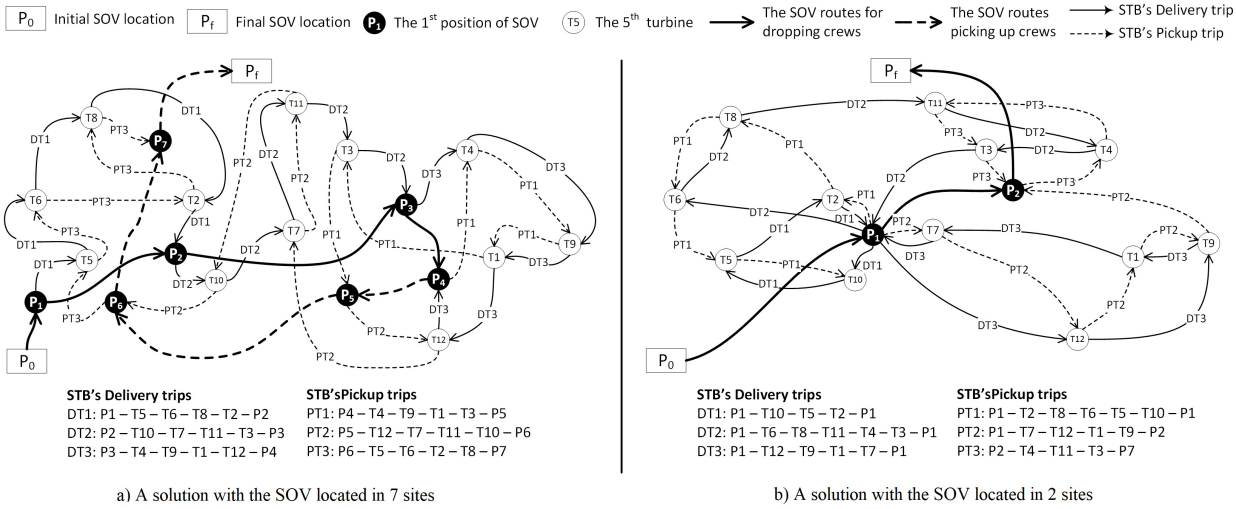

Figure 2: Examples of two feasible solutions for the problem

2.2. The mathematical model

Our mathematical model is based on a MINLP model. The following notation is used to describe the sets and parameters of the proposed model:

Sets and indices

I: set of turbines indexed by i and $n = |I|$
$C_i^I(x_i^I, y_i^I)$: the location of turbine $i \in I$.
$\tilde{m}$: the number of STB delivery trips
$\hat{m}$: the number of STB pick-up trips
m: the total number of STB trips (delivery and pick-up) where $m = \tilde{m} + \hat{m}$
s: index of STB trips, $s = \{1, \ldots, \tilde{m}, (\tilde{m}+1), \ldots, m\}$
l: index of SOV locations, $l = \{0, 1, \ldots, m+2\}$
$C_0^L(x_0^L, y_0^L)$: the initial SOV location, also denoted by P_0
$C_{(m+2)}^L(x_{(m+2)}^L, y_{(m+2)}^L)$: the final SOV location, also denoted by P_f
$C_l^L(x_l^L, y_l^L)$: the location of turbine $l(l = 1, \ldots, m+1\}$ that needs to be found.

$N_s \subseteq \{\{0\} \cup I \cup \{(n+1)\}\}$: the set of nodes that may be visited by the STB for each trip s; $s = 1, \ldots, m$. Here, node 0 represents the SOV location at the beginning of each STB trip, the unknown location is defined by $C_s^L(x_s^L, y_s^L)$, and node $(n+1)$ refers to the SOV location at the end of the STB trip, the unknown location of which is defined by $C_{(s+1)}^L(x_{(s+1)}^L, y_{(s+1)}^L)$.

Parameters

f^S: the SOV fixed cost when travelling from one point to another (€)
v^S: the travel cost of the SOV (€/km)
v^C: the travel cost of the STB (€/km)
e^S: the average speed of the SOV (km/hour)
e^C: the average speed of the STB (km/hour)
τ_i: the required time to maintain the turbine $i \in I$ (hour)
$\hat{\tau}$: the boarding time for crews and equipment to the STB from the SOV (hour)
$\tilde{\tau}_i$: the transfer time for crews and equipment from the STB to the turbine $i \in I$ (hour)
w_i: the weight of spare parts and equipment needed by the turbine $i \in I$ (kg)
$\hat{w}$: the total weight of parts and equipment that can be transported by the STB (load capacity)
ρ_i: the number of crew members needed to service the turbine $i \in I$
$\hat{\rho}$: the maximum number of crew members on board the STB (crew capacity)
$\ddot{\rho}$: the number of crew members available in the SOV
$\hat{t}_{\max}$: the latest time at which the last pick-up STB trip can return to the SOV
p^C: the penalty cost if the STB arrives at the SOV after $\hat{t}_{\max}$ (€/hour)
d_{ij}: the distance between turbines i and j, $(i,j) \in I$

Decision Variables

$C_l^L(x_l^L, y_l^L)$: the coordinate of SOV location at site $l; l = 1, \ldots, m+1$
T_l: the time when the SOV visit site $l; l = 0, \ldots, m+1$

$$U_s = \begin{cases} 1 & \text{if the STB is used for trip } s; s = 1, \ldots, m, \\ 0 & \text{otherwise} \end{cases}$$

$$X_{sij} = \begin{cases} 1 & \text{if the STB travels from node } i \text{ to } j, (i,j \in N_s) \text{ in trip } s; s = 1, \ldots, m, \\ 0 & \text{otherwise} \end{cases}$$

H_{si}: the time when the STB leaves node $i \in N_s$ in trip s; $s = 1, \ldots, m$

$$Y_l = \begin{cases} 1 & \text{if the SOV moves from position } l \text{ to } (l+1); l = 0, \ldots, m+1, \\ 0 & \text{otherwise} \end{cases}$$

The CLMRPOWF can be modelled through MINLP, which is expressed as follows:

$$\min Z = \max\{0, (T_{m+1} - \hat{t}_{\max})\} \cdot p^C + \sum_{l=0}^{l=m+1} \left(v^S \cdot D(C_l^L, C_{(l+1)}^L) + f^S \cdot Y_l\right) \tag{1}$$

$$+ \sum_{s=1}^{m} \sum_{j \in I} v^C \left[D(C_s^L, C_j^I) \cdot X_{s0j} + D(C_j^I, C_{(s+1)}^L) \cdot X_{sj(n+1)}\right] + \sum_{s=1}^{m} \sum_{i \in I} \sum_{j \in I} d_{ij} \cdot v^C \cdot X_{sij}$$

s.t.

$$D(C_l^L, C_{(l+1)}^L) = \sqrt{(x_l^L - x_{(l+1)}^L)^2 + (y_l^L - y_{(l+1)}^L)^2}, \quad \forall l = 0, 1, \ldots, (m+1) \tag{2}$$

$$D(C_s^L, C_j^I) = \sqrt{(x_s^L - x_j^I)^2 + (y_s^L - y_j^I)^2}, \quad \forall j \in I, s = 1, \ldots, m \tag{3}$$

$$D(C_j^I, C_{(s+1)}^L) = \sqrt{(x_j^I - x_{(s+1)}^L)^2 + (y_j^I - y_{(s+1)}^L)^2}, \quad \forall j \in I, s = 1, \ldots, m \tag{4}$$

$$X_{sij} \leq U_s, \quad \forall s = 1, \ldots, m; (i,j) \in N_s \tag{5}$$

$$\sum_{i \in I} X_{s0i} = U_s, \quad \forall s = 1, \ldots, m \tag{6}$$

$$\sum_{i \in I} X_{si(n+1)} = U_s, \quad \forall s = 1, \ldots, m \tag{7}$$

$$\sum_{s=1}^{\tilde{m}} \sum_{i \in N_s} X_{sij} = 1, \quad \forall j \in I \tag{8}$$

$$\sum_{s=1}^{\tilde{m}} \sum_{i \in N_s} X_{sij} = \sum_{s=\tilde{m}+1}^{m} \sum_{i \in N_s} X_{sij}, \quad \forall j \in I \tag{9}$$

$$\sum_{i \in N_s} X_{sij} = \sum_{i \in N_s} X_{sji}, \quad \forall j \in I, s = 1, \ldots, m \tag{10}$$

$$\sum_{i \in I} \sum_{j \in N_s} X_{sij} \cdot w_i \leq \hat{w}, \quad \forall s = 1, \ldots, \tilde{m} \tag{11}$$

$$\sum_{i \in I} \sum_{j \in N_s} X_{sij} \cdot \rho_i \leq \hat{\rho}, \quad \forall s = 1, \ldots, m \tag{12}$$

$$\sum_{s=1}^{\tilde{m}} \sum_{i \in I} \sum_{j \in N_s} X_{sij} \cdot \rho_i \leq \ddot{\rho} \tag{13}$$

$$H_{s(n+1)} \leq H_{(s')0} + M(1 - U_{(s')}), \forall s = 1, \ldots, (m-1), s' = s, \ldots, m \tag{14}$$

$$H_{s0} + \tilde{\tau}_i + D(C_s^L, C_j^I)/e^C - H_{sj} \leq M \cdot (1 - X_{s0j}), \quad \forall j \in I, s = 1, \ldots, m \tag{15}$$

$$H_{si} + \tilde{\tau}_i + d_{ij}/e^C - H_{sj} \leq M \cdot (1 - X_{sij}), \quad \forall s = 1, \ldots, m; (i,j) \in I \tag{16}$$

$$H_{si} + D(C_i^I, C_{(s+1)}^L)/e^C - H_{s(n+1)} \leq M \cdot (1 - X_{si(n+1)}), \quad \forall i \in I, s = 1, \ldots, \tilde{m} \tag{17}$$

$$H_{si} + \hat{\tau} + D(C_i^I, C_{(s+1)}^L)/e^C - H_{s(n+1)} \leq M \cdot (1 - X_{si(n+1)}), \quad \forall i \in I, s = (\tilde{m}+1), \ldots, m \tag{18}$$

$$\sum_{s=\tilde{m}+1}^{m} \sum_{j \in N_s} X_{sij} \cdot H_{si} - \sum_{s=1}^{\tilde{m}} \sum_{j \in N_s} X_{sij} \cdot H_{si} \geq (\tilde{\tau}_i + \tau_i) \cdot \sum_{s=1}^{\tilde{m}} \sum_{j \in N} X_{sij}, \quad \forall i \in I \tag{19}$$

$$T_0 + D(C_0^L, C_1^L)/e^S \leq T_1 \tag{20}$$

$$T_s + \hat{\tau} \leq H_{s0} + M(1 - U_s), \quad \forall s = 1, \ldots, \tilde{m} \tag{21}$$

$$T_s \leq H_{s0} + M(1 - U_s), \quad \forall s = (\tilde{m}+1), \ldots, m \tag{22}$$

$$T_{s+1} \geq H_{s(n+1)} + M(1 - U_s), \quad \forall s = 1, \ldots, m \tag{23}$$

$$H_{s0} + D(C_s^L, C_{(s+1)}^L)/e^S \leq T_{s+1}, \quad \forall s = 1, \ldots, m \tag{24}$$

$$D(C_l^L, C_{(l+1)}^L) \leq M \cdot Y_l, \quad \forall l = 1, \ldots, (m+1) \tag{25}$$

$$U_s = \{0, 1\}, \quad \forall s = 1, \ldots, m \tag{26}$$

$$X_{sij} = \{0, 1\}, \quad \forall s = 1, \ldots, m; (i, j) \in N_s \tag{27}$$

$$T_l \geq 0, \quad \forall l = 0, \ldots, m+1 \tag{28}$$

$$H_{si} \geq 0, \quad \forall s = 1, \ldots, m; i \in N_s \tag{29}$$

$$Y_l = \{0, 1\}, \quad \forall l = 0, \ldots, m+1 \tag{30}$$

$$C_l^L(x_l^L, y_l^L) \in \mathbb{R}^2, \quad \forall l = 1, \ldots, m+1 \tag{31}$$

The objective function (1) aims to minimise the total maintenance cost (Z). The first part of (1) defines the penalty cost, while the second part, which consists of three terms, represents the traveling and fixed costs. The first term refers to the traveling and the fixed costs of the SOV, while the remaining two represent the traveling cost of the STB. Note that the way the first term is defined with the inclusion of Constraints (25) is equivalent to $(v^S \cdot D(C_l^L, C_{(l+1)}^L) + f^S) \cdot Y_l$, which could have made this nonlinear problem even more complex. Constraints (2) calculate the Euclidean distance between two SOV locations. Constraints (3)–(4) determine the Euclidean distance between the SOV locations and the offshore turbines. Constraints (5)–(7) ensure that the routes of the STB are generated only for the selected trips, with start and end points at node 0 and node $(n+1)$, respectively. Constraints (8)–(9) guarantee that a turbine is maintained only once, i.e. the turbine is visited on one delivery trip and one pick-up trip only.

Constraints (10) state the flow conservation at each node and for each trip. Constraints (11) and (12) guarantee that the weight of equipment transported and the number of crew members on board do not exceed the STB's capacity. Constraints (13) ensures that the number of crew members needed to perform the maintenance activities on the selected turbines does not exceed the number of crew members available in the SOV. Constraints (14) ensure that the STB trip $(s+1)$ can be started only after the STB has returned to the SOV after trip s. Constraints (15) and (16) are used to determine the optimal time for the STB to leave a node, and the time needed for crew members to transfer from the STB to the turbines and the distances between two nodes are considered. Constraints (17) and (18) aim to obtain the optimal time for the STB to return to the SOV, taking into account the boarding times for the pick-up trips. Constraints (19) state that the time between the delivery and the pick-up must be greater than the time required to maintain the turbine.

Constraints (20)–(23) determine the optimal time for the SOV to arrive at the location where the STB will be deployed for each trip. Here, the SOV needs to be in a certain location before the STB leaves or returns to the SOV. For the delivery trips, boarding time is considered in Constraints (21). Constraints (24) state that the SOV may move if the STB has left the SOV. Constraints (25)

indicate whether the SOV location changes for each trip, turning Y_l to 1, and hence triggers the use of $D(C_l^L, C_{(l+1)}^L)$ and, consequently, the fixed cost in (1). Constraints (26)–(31) indicate the types of decision variable used.

In brief, the positions of the SOV when the STB trips start and end are defined as follows. At the beginning of STB trip s, the SOV is located at $C_s^L(x_s^L, y_s^L)$ (i.e., $s = l$). In the set N_s, this is represented by the start of STB trip s known as Node 0, while at the end of STB trip s, the SOV is located at $C_{(s+1)}^L(x_{(s+1)}^L, y_{(s+1)}^L)$ (i.e., $l = s+1$), which is indicated by Node $(n+1)$ in the set N_s. As the SOV locations $C_l^L(l = 1, \ldots, (m+1))$ are unknown, the distance between SOV locations $D(C_l^L, C_{(l+1)}^L)$ are variables. It is the same for the distance between SOV locations and turbines for each STB trip s ($D(C_s^L, C_j^I)$ and $D(C_j^I, C_{(s+1)}^L)$).

2.3. The mathematical model for the discrete model

As the mathematical model of the proposed problem (CLMRPOWF) is nonlinear and cannot be solved by the exact method (EM) using an off-the-shelf commercial optimiser, we propose the discrete model of the problem, which we refer to as the discrete location and maintenance routing problem for offshore wind farms (DLMRPOWF). The solutions generated by the EM (CPLEX) to solve the discrete problem are used as benchmarks for the proposed method when solving the continuous problem. In the discrete model, a set of potential sites (Q) for the locations of the SOV is introduced. The locations of the turbines to be maintained and the midpoints between any two turbines can be included in the set Q. The first and the last elements of the set Q are assigned to the initial location (P_0) and the final location (P_f) of the SOV, respectively. The notation used in the proposed model is the same as that adopted for the continuous problem. However, the variables $C_l^L(x_l^L, y_l^L)$ are now replaced by the following:

$$\hat{X}_{qrl} = \begin{cases} 1 & \text{if SOV travels from node } q \text{ to } r, (q, r \in Q) \text{ in the } l^{th} \text{ SOV location; } l = 1, \ldots, m+1, \\ 0 & \text{otherwise} \end{cases}$$

$$O_{ql} = \begin{cases} 1 & \text{if SOV is located in potential site } q \in Q \text{ in the } l^{th} \text{ SOV location; } l = 1, \ldots, m+1, \\ 0 & \text{otherwise} \end{cases}$$

The proposed mathematical model of the discrete problem is developed using MILP. The DLMRPOWF is modelled as follows:

$$\min Z = \max\{0, (T_{m+1} - \hat{t}_{\max})\} \cdot p^C + \sum_{l=0}^{l=m+1} \left[\left(f^S \cdot Y_l\right) + \left(\sum_{(q,r)\in Q} v^S \cdot d_{qr} \cdot \hat{X}_{qrl} \right) \right] \tag{32}$$

$$+ \sum_{s=1}^{m} \sum_{j\in I} v^C \left[X_{s0j} \sum_{q\in Q} d_{qj} \cdot O_{qs} + X_{sj(n+1)} \sum_{q\in Q} d_{jq} \cdot O_{q(s+1)} \right] + \sum_{s=1}^{m} \sum_{i\in I} \sum_{j\in I} d_{ij} \cdot v^C \cdot X_{sij}$$

s.t.

(5)–(14), (16), (19), (21)–(23), (26)–(30)

$$H_{s0} + \tilde{\tau}_i + \sum_{q \in Q} (O_{qs} \cdot d_{qj})/e^C - H_{sj} \leq M \cdot (1 - X_{s0j}), \quad \forall j \in I, s = 1, \ldots, m \tag{33}$$

$$H_{si} + \sum_{q \in Q} (O_{q(s+1)} \cdot d_{iq})/e^C - H_{s(n+1)} \leq M \cdot (1 - X_{si(n+1)}), \quad \forall i \in I, s = 1, \ldots, \tilde{m} \tag{34}$$

$$H_{si} + \hat{\tau} + \sum_{q \in Q} (O_{q(s+1)} \cdot d_{iq})/e^C - H_{s(n+1)} \leq M \cdot (1 - X_{si(n+1)}), \quad \forall i \in I, s = (\tilde{m}+1), \ldots, m \tag{35}$$

$$T_0 + \sum_{q \in Q} d_{0q} \cdot \hat{X}_{0q1}/e^S \leq T_1 \tag{36}$$

$$H_{s0} + \hat{\tau} \cdot U_s + \sum_{(q,r) \in Q} d_{qr} \cdot \hat{X}_{qr(s+1)}/e^S \leq T_{s+1}, \quad \forall s = 1, \ldots, \tilde{m} \tag{37}$$

$$H_{s0} + \sum_{(q,r) \in Q} d_{qr} \cdot \hat{X}_{qr(s+1)}/e^S \leq T_{s+1}, \quad \forall s = (\tilde{m}+1), \ldots, m \tag{38}$$

$$\sum_{(q,r) \in Q} \hat{X}_{qrl} = 1, \quad \forall l = 1, \ldots, (m+1) \tag{39}$$

$$\sum_{q \in Q} \hat{X}_{0q1} = 1 \tag{40}$$

$$d_{qr} \cdot \hat{X}_{qrl} <= Y_l \cdot M, \quad \forall (q,r) \in Q, l = 1, \ldots, (m+1) \tag{41}$$

$$\sum_{q \in Q} \hat{X}_{qrl} = \sum_{q \in Q} \hat{X}_{rq(l+1)}, \quad \forall r \in Q, l = 1, \ldots, m \tag{42}$$

$$O_{ql} \leq \sum_{r \in Q} \hat{X}_{rql}, \quad \forall q \in Q, l = 1, \ldots, (m+1) \tag{43}$$

$$\sum_{q \in Q} O_{ql} = 1, \quad \forall l = 1, \ldots, (m+1) \tag{44}$$

$$\hat{X}_{qrl} = \{0, 1\}, \quad \forall (q,r) \in Q, l = 1, \ldots, (m+1) \tag{45}$$

$$O_{ql} = \{0, 1\}, \quad \forall q \in Q, l = 1, \ldots, (m+1) \tag{46}$$

Similar to that of the continuous model, the objective function of the discrete model is to minimise the total maintenance cost. Constraints (33)–(35) determine the times at which the STB leaves and returns to the SOV. Constraints (36)–(38) track the time of each SOV trip. Constraints (39) and (40) restrict the movement of each SOV trip. Constraints (41) have the same function as Constraints (25). The flow conservation of the SOV is expressed by Constraints (42). Constraints (43) and (44) determine the optimal location of the SOV. Constraints (45) and (46) state that variables $\hat{X}_{qrl}$ and O_{ql} are binary.

Constraints (19) and the objective function (32) make the problem nonlinear due to the product of two decision variables (integer/real and binary). In the implementation, the problem is transformed into a linear problem (MILP) using the common scheme of introducing new non-negative decision variables and the necessary constraints. The transformed model can then be solved using a commercial solver such as CPLEX, Xpress or Gurobi.

3. The Proposed Algorithm for the CLMRPOWF

The GA is one of the most powerful population-based metaheuristics used to solve complex combinatorial and optimisation problems. This method is inspired by the theory of evolution, as the concepts of natural selection, reproduction, genetic heritage, and mutation are invoked (Holland, 1992). Since the GA is population-based, it is powerful for diversification purposes but can suffer from intensification. To overcome this weakness, hybrid genetic algorithms (HGAs) are usually implemented when a local search is embedded within the GA. This is performed either for every chromosome (which can be time-consuming), or for certain defined chromosomes, or following a certain number of generations, which can be decided periodically or adaptively.

3.1. Algorithm overview

Figure 3 illustrates an example of the continuous LRP. The figure shows that the SOV may travel from the initial location P_0 to P_1 till P_{m+1}, and will ultimately end up at the final location P_f. Here, the SOV needs to be located before the STB is deployed to access the turbines for each trip. Therefore, there will be $(m+1)$ SOV locations in the plane that need to be determined, where the location P_{m+1} is the SOV location for the last pick-up trip of the STB. It is noted that the coordinates of the SOV locations (i.e. P_l, $l = 1, \ldots, m+1$) may be identical, as the SOV may not move to avoid additional fixed costs. It is assumed that once the STB returns to the SOV, the STB will leave the SOV from the same SOV location. For example, in Figure 3, the STB will return from the first delivery trip to the SOV located at P_2, then, with the SOV located still at P_2, will embark on the second trip.

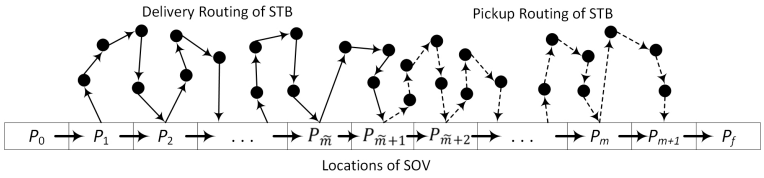

Figure 3: The LRP for the SOV and the STB

For our proposed method, a HGA is developed in which the GA is designed to find the best locations for the SOV. Once the location of the SOV for each trip is known, the delivery and pick-up routes of the STB are constructed by a metaheuristic based on VNS. The SOV locations are then improved using a new iterative mechanism based on the Weiszfeld algorithm and the end-point algorithm originally proposed by Salhi (1987) and Salhi and Nagy (2009). As previously mentioned, we refer to our new mechanism as the MEPA. The main objective of the MEPA is to find, for the given end-points of the routes found so far, the optimal SOV locations in the plane to reduce the total traveling cost. The main steps of the proposed HGA, which consists of two phases, are presented in Algorithm 1.

Algorithm 1 Procedure Hybrid GA

1: **Phase 1:** HGA
2: Set $t = 0$ and the parameter λ
3: Create an initial population $G(0)$ representing the SOV locations
4: **for each** individual in $G(0)$ **do**
5: Generate STB routes using the mini VNS algorithm
6: Apply the MEPA algorithm
7: Evaluate the fitness values (Z and Z^{bm})
8: **end for**
9: Classify the population (α_1, α_1 and α_3)
10: **repeat**
11: Set $G'(t) \leftarrow G(t)$ and $\theta =$ false
12: **if** A random number in the range [0,1] $< \lambda$ **then** $\theta =$ true
13: Select pairs of parent based on the θ value ($G''(t) \leftarrow G'(t)$)
14: Perform the crossover on $G''(t)$ based on crossover rate p_c
15: Conduct the mutation on $G''(t)$ based on mutation rate p_m
16: Implement the mini VNS to generate STB routes for each individual in $G''(t)$
17: **if** $\theta =$ true **then**
18: Apply the MEPA algorithm for each individual in $G''(t)$
19: **end if**
20: Evaluate the fitness values for each individual in $G''(t)$
21: **if** A random number in the range [0,1] $< (p^e)$ **then**
22: Replace the worst w individuals in $G''(t)$ by random ones.
23: **for each** random individual **do**
24: Implement the mini VNS to obtain STB routes
25: Apply the MEPA algorithm
26: Evaluate the fitness values (Z and Z^{bm})
27: **end for**
28: **end if**
29: Classify the population (α_1, α_1 and α_3)
30: $G(t+1) \leftarrow G''(t)$
31: $t = t + 1$
32: **until** the termination condition is met
33: **Phase 2:** Post-optimization
34: Take γ best distinct solutions from $G(t)$ denoted as G^*.
35: **for each** individual in G^* **do**
36: Implement the full VNS algorithm to generate STB routes
37: Apply the MEPA algorithm
38: Evaluate the fitness value (Z)
39: **end for**
40: Take the best solution of the problem.

In the first phase of Algorithm 1, an iterative HGA is designed where Parameter λ is introduced to determine whether the proposed MEPA needs to be applied to chromosomes in a certain generation. When the MEPA is not executed, the process will be relatively faster, but this occurs at the expense of a reduction in the quality of the solutions generated. Note that when $\lambda = 1$, the MEPA will be executed for all generations, while when $\lambda = 0$, the MEPA will not be used at all. Phase 1 of Algorithm 1 deals with the discrete problem (DLMPRPOWF) only if $\lambda = 0$ and if the MEPA in Line 6 of Algorithm 1 is not executed. In other words, the locations of the SOV will be based on the potential sites Q. In the first phase, a mini VNS is applied instead of the full VNS to

reduce the computational time. The mini VNS relies on one shake with one neighbourhood structure ($k_{\max} = 1$) and only uses one iteration ($i_{\max} = 1$). The second phase is the post-optimisation phase, which consists of the selection of γ best distinct solutions from Phase 1, the full VNS and the MEPA algorithm. Explanations of these two algorithms are provided in Subsections 3.4 and 3.5, respectively.

Once a chromosome (i.e. the locations of the SOV) is generated, the delivery and pick-up routes of the STB are obtained using the proposed VNS. The fitness value of each chromosome is calculated based on the total traveling costs of the SOV and STB, together with the penalty cost incurred if the STB returns to the SOV later than $t_{\max}$. For each chromosome, we record both the final total cost (Z) and the total cost before the MEPA is executed (Z^{bm}). These two costs are introduced to classify the population that will be used for parent selection (Subsection 3.3).

3.2. Chromosome encoding and initial population

In the GA, we represent the chromosome using $(m+1)$ columns, from P_1 to P_{m+1}. The columns represent the locations of the SOV at the start and the end of each STB trip. An initial population of N chromosomes is generated where each chromosome is constructed based on the potential sites (Q) that have been defined. The set of sites may consist of the locations of the turbines (I) and the midpoints between the turbines (K) together with the SOV's initial and final locations (P_0 and P_f, respectively). In mathematical terms, $Q = \{P_0\} \cup I \cup K \cup \{P_f\}$. Figure 4 illustrates the chromosome representation when six trips take place ($m = 6$). At the beginning of the first STB trip, the SOV is located at site q_4 whereas at the end of the final trip, the SOV is located at site q_2. Note that the SOV stays at site q_{12} during Trips 2 and 3. Note that $S = \{q_1, q_2, \ldots, |S|\}$.

q_4	q_{12}	q_{12}	q_{10}	q_9	q_2	q_2
$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6 = m$	$m + 1$

Figure 4: The chromosome representation

Algorithm 2 describes the procedure to generate the initial population. The chromosome is constructed based on the sets I and K. When selecting any SOV location, it should not be too far from previous location. Therefore, a parameter (A_q) that defines the average distance from a potential site ($q \in Q$) to the other sites in Q is introduced. The maximum number of sites used in each chromosome (β) is randomly generated. Line 15 of Algorithm 2 controls the number of the sites used in a solution. Here, if the number of sites in a chromosome has reached β, the next gene (site) is the same as the previous one generated. There is also the probability of using the site (gen) that has been previously allocated, which is represented in Lines 18-19 of Algorithm 2.

3.3. Genetic operators

The following genetic operators, which include parent selection, crossover, mutation and the injection operator that deals with the immigration effect, are adopted.

Algorithm 2 Procedure Initial Population Generation

1: Define set Q and φ value
2: Calculate A_q, the average distance from site $q \in Q$ to other sites
3: **for** $i = 1$ to N **do**
4: Generate a random integer number β in the range $[1, m]$
5: Set $s = 0$
6: **for** $j = 1$ to m **do**
7: Generate a random number π in the range $[0, 1]$
8: **if** $j = 1$ **then**
9: **if** $\pi < \varphi$ **then** Choose randomly site $\hat{q}$ from set I
10: **else** Choose randomly site $\hat{q}$ from set K
11: **if** $d(P_0, \hat{q}) > A_{P_0}$ **then** Go back to Line 6.
12: **else** Set $P_j \leftarrow \hat{q}$
13: $s = s + 1$
14: **else**
15: **if** $s = \beta$ **then**
16: Set $P_j = P_{(j-1)}$
17: **else**
18: **if** A random number in the range $[0, 1] < 0.5$ **then**
19: Set $P_j = P_{(j-1)}$
20: **else**
21: **if** $\pi < \varphi$ **then** Choose randomly site $\hat{q}$ from set I
22: **else** Choose randomly site $\hat{q}$ from set K
23: **if** site $\hat{q}$ has been chosen **then** Go back to Line 21
24: **if** $d(P_{(j-1)}, \hat{q}) > A_{P_{j-1}}$ **then** Go back to Line 21
25: $s = s + 1$
26: **end if**
27: **end if**
28: **end if**
29: **end for**
30: **end for**

3.3.1. Parents selection

This genetic operator guides the selection of pairs of parents for mating and reproducing a number of offspring, with the aim of yielding better chromosomes. A selection process based on the survival of the fittest is designed to ensure a higher likelihood that good chromosomes will be chosen. In the proposed GA, we do not consider all chromosomes to contribute equally to solving the problem. At each generation, therefore, we classify the population into three categories based on the fitness value of each individual using Z^{bm}, and Z. Based on Z^{bm}, individuals are classified into classes α_1^{bm}, α_2^{bm} and α_3^{bm} (α^{bm} classification), whereas based on Z, they are categorised into classes α_1^{am}, α_2^{am}, and α_3^{am} (α^{am} classification). An individual classified as α_1^{am} has a good fitness value, whereas an individual classified as α_3^{am} has a relatively less attractive fitness value. An individual may receive different Z^{bm} and Z classifications. For example, Individual 1 may be classified as α_1^{bm} and α_2^{am}. This classification is used when selecting a pair of parents for breeding. The number of individuals for each classification is equally distributed. This kind of classification within GA was shown to be promising when applied to a class of continuous location problems (Salhi and Gamal, 2003) and also in routing problems (Salhi and Petch, 2007).

In the implementation, the selection of the parents is performed by considering the value of θ in Algorithm 1 (θ is true if the MEPA is used for chromosomes in a certain generation; otherwise, θ is false). Algorithm 3 summarises this flexible parent's selection operator. Here, we first randomly choose the type of population classification where class α_1 (α_1^{bm} or α_1^{am}) has a higher probability to be selected (i.e. $p(\alpha_1) = 0.6, p(\alpha_2) = 0.3$ and $p(\alpha_3) = 0.1$). We then randomly choose ψ individuals from this classification. The tournament selection is then implemented wherein the best individual from the ψ individuals is chosen as a parent based on its fitness value. We set the value of ψ to 5 which is based on our preliminary experiments.

Algorithm 3 The Parent Selection Operator

if θ = false **then**
  Select a pair of parents based on the α^{bm} classification
else
  Generate randomly an integer number (ω) in the range [1,3]
  if $\omega = 1$ **then**
    Select a pair of parents using the α^{am} classification
  else if $\omega = 2$ **then**
    Select a pair of parents based on the α^{bm} classification
  else
    Select one parent using the α^{am} classification and the other using the α^{bm} classification
  end if
end if

3.3.2. Crossover and mutation

The crossover operator aims to promote diversity in the new offspring while producing a new population that is able to survive in the next generation based on the good qualities of the parents. Here, one-point and two-point crossovers are implemented with a crossover rate (p_c). The crossover points are randomly determined. The operator is performed by cutting the columns of the matrices representing the two parents and separating the genes to produce two new individuals (Child 1 and Child 2), as illustrated in Figure 5.

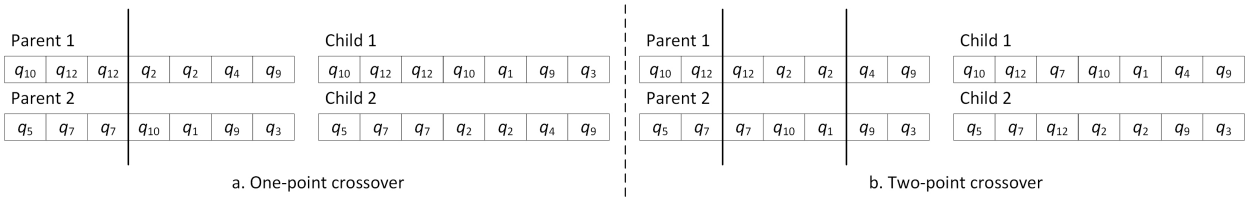

Figure 5: Crossover operator.

The mutation process is randomly performed with a probability p^m (mutation rate) for each individual. This is conducted by randomly modifying some genes to produce new offspring, avoid local optima and diversify the population. An individual is chosen to be mutated, and then a gene (a SOV location) is randomly selected, say A_q. This is replaced by a new site taken from set Q,

which is not too distant from A_q. Here, we consider the threshold as the average distance of the potential sites from A_q.

3.3.3. Injection operator and termination criteria

For each generation, we inject new chromosomes with a probability (p^e). In real life, this represents the positive effect immigrants can have on the economy and is intended to maintain the diversity level of the population by replacing a number of individuals in the current population with random individuals (immigrants) at each generation. When this operator is applied, the worst w individuals in the current population are replaced by these randomly generated ones. Here, w is set to a small value (i.e. $0.2N$) in order to guide the search while introducing only small perturbations (Yang, 2008). Algorithm 2 of the initial population generation procedure is used to construct these random individuals.

There are several termination (stopping) criteria that can be implemented, including the number of generations, the convergence of fitness values, and no improvement in the best fitness value for a certain number of successive generations. In our proposed method, we use the number of generations (NG) that is determined by the size of the problem. In this case, the value of NG is based on the number of turbines and the number of trips, where $NG = \lfloor N/3 \rfloor$ and $N = \min\{100, m \cdot |I|\}$.

3.4. VNS algorithm for generating the STB routes

VNS was first introduced by Mladenović and Hansen (1997). Three key steps in the VNS include shaking, improvement, and neighbourhood change procedures. The shaking step is used to perturb an existing solution to resolve local minima traps. The improvement step aims to find local optimality for a given solution configuration that can be performed using a single local search as in the basic VNS or a local search engine like a variable neighbourhood descent (VND) procedure as in the general VNS. The neighbourhood change step aims to explore the solution space that can be performed using several methods including sequential, cyclic, pipe, and skewed neighbourhood change steps. For more information on VNS and its variants, see Hansen et al. (2010) and Hansen et al. (2017), and for an overview of heuristic search, including VNS, see Salhi (2017).

The key steps of the proposed VNS are presented in Algorithm 4. It is categorised as a General VNS, according to the classification provided by Hansen et al. (2017), as the improvement step is made up of more than one local search, forming a local search engine. In the first step of the Algorithm 4, groups of turbines are formed for each delivery and pick-up trip. Next, the initial STB routes are generated based on the groups constructed. In the VNS algorithm, the shaking procedure is performed by executing an operator from a neighbourhood structure (N_k). In the improvement step, a local search engine known as the multi-level heuristic, as developed by Salhi and Sari (1997), is implemented. This multi-level local search is similar in principle to a sequential VND. In the neighbourhood change step, a sequential strategy is used. In Phase 1 of the main HGA procedure (Algorithm 1), when executing the VNS to generate the STB routes, the values of $k_{\max}$ and $i_{\max}$ are set to 1 to reduce the computational time. We refer to this type of VNS as a mini VNS

due to its simplicity. In the post-optimisation phase of Algorithm 1, the full VNS is executed with $i_{\max} = 25$ and $k_{\max} = 7$, which is determined based on the preliminary computational experiments considering the quality of solutions generated and the computational time required.

Algorithm 4 General VNS for generating the STB's routes

```
Form groups of turbines for each delivery and pickup trip
Generate initial STB routes based on the groups of turbines formed
Define neighborhood structures N_k, for k = 1, ..., k_max
Set i = 1
repeat
    Set k = 1
    while k ≤ k_max do
        Shaking: Generate randomly a point x' from the k^th neighborhood of x (x' ∈ N_k(x))
        VND: Apply a multi-level local search engine to obtain the best neighborhood x'' (Subsection 3.4.3).
        if the local optimum x'' is better than the incumbent x then
            Update x ← x'' and k = 1
        else
            Update k = k + 1.
        end if
    end while
    Update i = i + 1
until i = i_max
```

3.4.1. Group construction and initial solution

We construct groups of offshore turbines based on the number of delivery or pick-up trips (m) and the SOV locations ($P_j, j = 1, \ldots, \tilde{m}$ for the delivery trips and $j = \tilde{m}+1, \ldots, m$ for the pick-up trips). In this case, $m - \tilde{m} = \hat{m}$. Initially, each SOV location is grouped with its nearest turbine. Next, each turbine that has not been allocated is assigned to the nearest group (based on its distance from the closest member). This also considers the number of crew members and the weight of the equipment allocated to each group since the STB has a limited capacity. A turbine can be allocated to the second-nearest group to meet these constraints. If this procedure is not able to construct groups that include all the turbines, the modified generalized assignment problem (GAP) is adopted. In the modified GAP, the allocation of a turbine to its respective SOV locations (Y_{ij}) is considered as the binary decision variable. This modified GAP is given in the following mathematical model.

$$\max \sum_{i \in I} \sum_{j=1}^{\tilde{m}} (Y_{ij} \cdot (d(i,j) + d(i,j+1))/2) \tag{47}$$

s.t.

$$\sum_{j=1}^{\tilde{m}} Y_{ij} = 1, \forall i \in I \tag{48}$$

$$\sum_{i \in I} Y_{ij} \cdot \rho_i \leq \hat{\rho}, \forall j = 1, \ldots, \tilde{m} \tag{49}$$

$$\sum_{i \in I} Y_{ij} \cdot w_i \leq \hat{w}, \forall j = 1, \ldots, \tilde{m} \tag{50}$$

$$Y_{ij} = \{0, 1\}, \forall i \in I, j = 1, \ldots, \tilde{m} \tag{51}$$

The modified GAP model is an integer linear programming (ILP) model, which can be optimally solved using any suitable ILP solver such as CPLEX or Gurobi. Once the turbines have been grouped for each delivery and pick-up trip, the initial STB routes for these trips are generated. The initial routes are obtained by implementing Gillett and Miller (1974) sweep algorithm for each group (intra-route). Each route is then improved by executing Lin (1965) 2-opt algorithm. This solution is then used as the initial solution (x) for the VNS algorithm.

3.4.2. Neighborhood Structures

This subsection presents the neighbourhood structures that are applied in the shaking phase. We examined seven neighbourhood structures ($N_k, k = 1, \ldots, k_{max}$) as follows:
The first neighbourhood (N_1) is the 1-0 shift in which a random turbine from a randomly chosen route is selected. This turbine is then inserted into the route that yields the smallest total cost. The position of the turbine in the chosen route also needs to be determined.
In the second neighborhood (N_2), the 1-1 interchange is implemented, generating a feasible solution by swapping a pair of turbines from two routes. This procedure begins by taking a random turbine from a randomly chosen route and swapping it with another turbine from another route to minimises the total cost.
Note that the local searches LS1 and LS3 described in Subsection 3.4.3 are based on the neighbourhood structures N_1 and N_2, respectively except that in the N_1 and N_2, one turbine is first randomly selected only whereas in the local searches their respective neighbourhoods are entirely examined leading to the adoption of the best improvement strategy.
The third and the fourth neighbourhoods (N_3 and N_4) are the random 1-0 shift and the random 1-1 interchange, respectively. These are similar to N_1 and N_2, respectively, except that the random turbine from a randomly selected route is inserted into a random feasible route or swapped with a random turbine from a randomly chosen route.
The fifth neighbourhood (N_5) is the random 1-1-1 interchange, which considers three routes simultaneously. A random turbine is first taken from a randomly chosen route. This turbine is then allocated to a second randomly selected route, while a random turbine from the second route is assigned to a third randomly chosen route. Finally, a random turbine from the third route is inserted into the first route. This process is repeated until a feasible solution is found. The basis of this mechanism was initiated by Salhi (1987) and is known as 'perturb'.
The sixth and seventh neighbourhoods (N_6 and N_7) are the random 2-1 and 2-2 interchanges, respectively. These neighbourhoods are similar to N_4 except in the number of turbines that need to be swapped.

3.4.3. Multi-level local search engine

This subsection describes the multi-level engine that is applied in the improvement/intensification phase. Seven local searches are adopted to make up the multi-level local search heuristic. Based on

the complexity of the local searches, we adopt the following order: LS1, the 1-insertion inter-route; LS2, the 2-opt inter-route; LS3, the 1-1 swap/interchange inter-route; LS4, the 2-opt intra-route; LS5, the 1-1 interchange intra-route; LS6, the 1-insertion intra-route; and LS7, the 2-insertion intra-route.

The best improvement strategy is used for each local search. Whenever an improvement is identified by any local search (LS2 to LS7), the process is repeated, starting from LS1. This multi-level local search heuristic, developed by Salhi and Sari (1997), acts in a similar way to the variable neighbourhood descent outlined by Hansen et al. (2010).

In LS1 (the 1-insertion inter-route), each turbine from a route is removed from its position, and an attempt is made to allocate it elsewhere in another route. If this is feasible and considered to be the best improvement, the selected turbine is then permanently moved. The insertion process is repeated until no further improvement can be found. In LS6 (the 1-insertion intra-route), we move a turbine position to another position in a route to achieve a better solution, while in LS7 (the 2-insertion intra-route), two consecutive turbine positions are checked for possible insertion as one block within a route to reduce the traveling cost.

The 2-opt intra-route (LS4) proposed by Lin (1965) works by removing two non-adjacent arcs and adding two new arcs within a route. An exchange is accepted if the resulting total cost is better than the previous one. The exchange process is continued until no improvement can be found. In the 2-opt inter-route (LS2), two routes are considered in which each of the two arcs belong to a different route. This operator reverses directions of the corresponding affected path of each route, as illustrated in Figure 6, where arcs a and b are replaced by arcs a' and b', respectively. In this particular problem, the exchange needs to be performed for two adjacent routes. For example, the exchange cannot be carried out for the routes started at P_1 and P_3 because it will affect the sequence of the trips and the movement of the SOV.

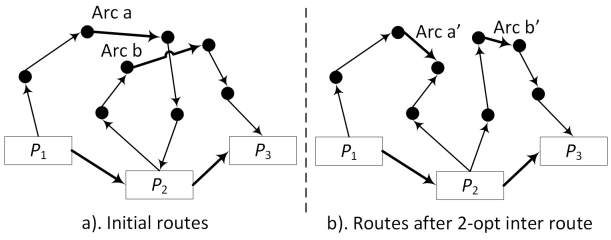

Figure 6: The 2-opt inter-route operator.

Two types of swap/interchange procedure are implemented: the 1-interchange intra-route (LS5) and the 1-interchange inter-route (LS3). The interchange intra-route aims to improve the route configuration by swapping the positions of a pair of turbines within the same route. In the interchange inter-route, the pair of turbines belong to different routes.

These local searches all have $O(m^2)$ time complexity, including LS7, as it uses two consecutive locations as one block and reinserts the block within the route. However, in practice, the inter-

routes-based ones generally consume relatively less computational time than others since they are based on individual routes. In brief, in the VNS method, a random movement is performed in the neighbourhood structures to perturb the incumbent solution with the aim of escaping from the local optima. In the local search, a systematic movement (rather than random movement) that explores the whole of the chosen neighbourhood is implemented instead in order to determine the best solution for that neighbourhood (i.e. the local minimum).

3.5. The modified end-point algorithm (MEPA)

The MEPA is based on the Weiszfeld iterative procedure, which guarantees ϵ optimality for the case of one facility. In other words, the procedure terminates when the difference between the cost of two successive solution configurations is less than ϵ with each update requiring just $O(1)$. The main objective of this algorithm is to find the SOV locations in the plane based on the SOV locations obtained from the potential sites (Q). Let $\tilde{m}$ and $\hat{m}$ be the number of delivery and pickup routes generated, respectively, where $m = \tilde{m} + \hat{m}$. Let R^j be the j^{th} route where $j = 1, \ldots, \tilde{m}, (\tilde{m}+1), \ldots, m$. Let R_s^j and R_e^j be the start and the end point of route j respectively with $R_s^j(a_s^j, b_s^j)$ and $R_e^j(a_e^j, b_e^j)$ known. Figure 7 illustrates SOV and STB routes where the SOV locations at P_0 and P_f are fixed. Note that MEPA relies on the Weiszfeld recursive technique to determine the new optimal location based on the demand nodes, which consists of the end-points of the routes and the two closed SOV locations. Here, the SOV locations at P_1 and P_2 until P_{m+1} are considered one at a time in a recursive manner to reduce the traveling cost.

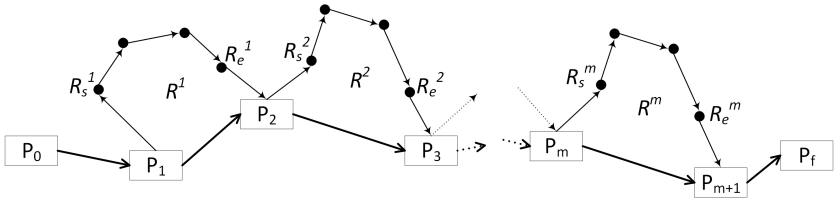

Figure 7: The illustration of SOV and STB routes

The new SOV location (P_j) is determined by considering the locations of (P_{j-1}), (P_{j+1}), R_e^{j-1} and R_s^j. However, the points R_e^{j-1} and R_s^j are not taken into account for the location of P_1 and $P_{(m+1)}$ respectively as these SOV locations consist of 3 arcs only as illustrated in Figure 7. The general model to find the optimal locations $(P_1, \ldots, P_{(m+1)})$ for locating the SOV is as follow:

$$\min \sum_{j=1}^{m+1} \left[w_2 \cdot (d(P_j, P_{j-1}) + d(P_j, P_{j+1}))\right] + \left[w_1 \cdot d(P_1, R_s^1)\right] \tag{52}$$
$$+ \sum_{j=2}^{m} \left[w_1 \cdot \left(d(P_j, R_e^{j-1}) + d(P_j, R_s^j)\right)\right] + \left[w_1 \cdot d(P_{m+1}, R_e^m))\right]$$

where $w_2 >> w_1$. Here, w_2 is represented by the fixed and variable costs of the SOV whereas w_1 the variable cost of the STB. The decision variables will be $P(x_j, y_j) \in \mathbb{R}^2$ and the Euclidean distance is used as follows:

$d(P_j, P_{j-1}) = \sqrt{(x_j - x_{(j-1)})^2 + (y_j - y_{(j-1)})^2}$, $d(P_j, P_{j+1}) = \sqrt{(x_j - x_{(j+1)})^2 + (y_j - y_{(j+1)})^2}$, $d(P_j, R_e^{j-1}) = \sqrt{(x_j - a_e^{j-1})^2 + (y_j - b_e^{j-1})^2}$ and $d(P_j, R_s^j) = \sqrt{(x_j - a_s^j)^2 + (y_j - b_s^j)^2}$.

The model can be solved using an NLP solver. However, this is time-consuming and inefficient. The approximated approach that we develop is based on the iterative process of Weiszfeld (1937). This algorithm, MEPA, is presented in Algorithm 5. In the algorithm, the total cost (Z^{bm}) and the set $P_j (j = 1, \ldots, m+1)$ and $(R_s^j, R_e^j), j = 1, \ldots, m$ are required. The new location, $\tilde{P}_j(\tilde{x}, \tilde{y})$, in the plane is based on the current location $\bar{P}_j(\bar{x}, \bar{y})$ and iteratively obtained using the Weiszfeld equations represented by (53) and (54) as follows:

$$\tilde{x}_j = \frac{\frac{w_2 \cdot (\bar{x}_{j-1} + \bar{x}_{j+1}) + w_1 \cdot \left(a_s^j + a_e^{j-1}\right)}{d(P_j, P_{j-1}) + d(P_j, P_{j+1}) + d(P_j, R_e^{j-1}) + d(P_j, R_s^j)}}{\frac{w_2}{d(P_j, P_{j-1})} + \frac{w_2}{d(P_j, P_{j+1})} + \frac{w_1}{d(P_j, R_e^{j-1})} + \frac{w_1}{d(P_j, R_s^j)}} \tag{53}$$

$$\tilde{y}_j = \frac{\frac{w_2 \cdot (\bar{y}_{j-1} + \bar{y}_{j+1}) + w_1 \cdot \left(b_s^j + b_e^{j-1}\right)}{d(P_j, P_{j-1}) + d(P_j, P_{j+1}) + d(P_j, R_e^{j-1}) + d(P_j, R_s^j)}}{\frac{w_2}{d(P_j, P_{j-1})} + \frac{w_2}{d(P_j, P_{j+1})} + \frac{w_1}{d(P_j, R_e^{j-1})} + \frac{w_1}{d(P_j, R_s^j)}} \tag{54}$$

where (a_e^{j-1}, b_e^{j-1}) and (a_s^j, b_s^j) are not considered for $(\tilde{x}_0, \tilde{y}_0)$ and $(\tilde{x}_{m+1}, \tilde{y}_{m+1})$ respectively.

Algorithm 5 The MEPA Algorithm

Require: The total cost Z^{bm}
Require: Set $P_j (j = 1, \ldots, (m+1))$ and $(R_s^j, R_e^j), j = 1, \ldots, m$
1: $Z = Z^{bm}$
2: **for** $j = 1$ to $(m+1)$ **do**
3: Set $\bar{P}_j \leftarrow P_j$
4: Use the iterative Weiszfeld equations to find $\tilde{P}_j$
5: **if** $d(\bar{P}_j, \tilde{P}_j) > \epsilon$ **then**
6: Calculate the leaving times of STB and SOV for each visited node based on the new site $\tilde{P}_j$
7: Determine the new total maintenance cost z'
8: **if** $z' < Z$ **then** Update $\bar{P}_j \leftarrow \tilde{P}_j$ and $Z = z'$
9: Go back to Line 3
10: **end if**
11: Set $P_j \leftarrow \bar{P}_j$
12: **end for**
13: Use VNS (Alg. 4) to generate the STB routes based on the new P_j
14: Update the total cost Z
15: **if** some (R_e^j, R_s^j) are changed **then** Go back to Line 1
16: **return** Z and $P_j (j = 1, \ldots, (m+1)))$

Once the location of $\tilde{P}_j$ has been changed ($d(\bar{P}_j, \tilde{P}_j) > \epsilon$), the times at which the STB and SOV leave the nodes are determined, and the new total maintenance cost z' is calculated. It should be noted that the sequence of STB routes is unchanged for each trip. If the cost is improved, then both the SOV location and the cost are updated. This procedure is repeated until the new location of $\tilde{P}_j$ is not changed by considering the parameter ϵ. In the proposed HGA method (Algorithm 1), ϵ is set to 0.0001 and 0.000001 for Phases 1 and 2, respectively. When all new SOV locations are obtained, the proposed VNS (Algorithm 4) is executed to construct the routes of the STB based on

the new SOV locations. To avoid evaluating unnecessary computations, we introduce the following reduction scheme. If, in the new routes, the configuration of (R_e^j, R_s^j) is changed (i.e. the end points of route j are changed), the MEPA is repeated from the beginning; otherwise, there is no change in either the location or the end points, and therefore, there is no need to continue.

3.6. Benchmarking against two other heuristic algorithms for the CLMRPOWF

To evaluate the performance of the proposed genetic operators designed in the GA, we develop two heuristic-based approaches: an evolutionary algorithm (i.e. the PSO) and a multi-start method (MS). As we aim to assess our GA-based metaheuristic, for consistency, both the PSO and the MS will incorporate the same VNS and MEPA implementations used by the HGA. For simplicity, the proposed PSO and MS are referred to as the hybrid PSO (HPSO) and the hybrid MS (HMS), respectively.

3.6.1. The hybrid particle swarm optimisation

The PSO is considered an evolutionary-type approach and was introduced by Kennedy and Eberhart (1995). The PSO has shown to be especially promising in solving (i.e. continuous) optimisation problems. For instance, the PSO has successfully addressed continuous location problems, including the continuous location-allocation problem (Ghaderi et al., 2012), the p-centre problem (Rabie et al., 2013), and the continuous p-median. PSO is developed so that each particle (p) of the swarm evaluates its new position ($X_p(k+1)$) at iteration $(k+1)$ with respect to its current position $X_p(k)$, its best position X_p^{PB}, its updated velocity $V_p(k+1)$, and the global best position of the entire swarm (X^{GB}). This is defined as:

$$X_p(k+1) = X_p(k) + V_p(k+1) \tag{55}$$

$$V_p(k+1) = w^k \cdot V_p(k) + c_1 \cdot r_1 \cdot (X_p^{PB} - X_p(k)) + c_2 \cdot r_2 \cdot (X^{GB} - X_p(k)) \tag{56}$$

w^k refers to the inertia weight and is a non-increasing function of the iteration number k. r_1 and r_2 are uniformly distributed random numbers chosen at each iteration from the range [0,1]; c_1 and c_2 are acceleration parameters linked to the current best and global best positions, respectively. The search begins with an initial solution, and the swarm keeps moving by updating its position based on the two above equations until a stopping criterion is met.

Similar to the proposed HGA, the HPSO comprises two phases. In the first phase, the population of particles is generated based on Algorithm 2. The velocity for each particle is randomly generated by considering the locations of the turbines. Once the position of a particle is updated, based on Equations (55) and (56), the mini VNS and MEPA algorithms are executed. The parameters used in the HPSO are set as follows: r_1 and r_2 are randomly generated in the range [0,1]; c_1 is randomly generated in the range [0,4], while $c_2 = 4 - c_1$; and the inertia weight (w) is systematically reduced over the iterations where $w^{k+1} = 0.9 \cdot w^k$. In the second phase, the γ best distinct solutions, such as the HGA method, are used.

3.6.2. The hybrid MS

We also provide a comparison using the HMS that we developed. A multi-start technique has also been implemented to solve the continuous problem (Cooper, 1964; Redondo et al., 2009; Brimberg et al., 2014). The proposed HMS is designed based on an iterative process that integrates the VNS algorithm and the MEPA algorithms. The HMS also consists of two phases. In the first phase, the SOV locations are randomly selected from the potential sites (Q), which are constructed based on the turbine sites and the midpoints between the turbines. This is followed by the mini VNS and the MEPA. This process is repeated until the stopping criteria are met. Then, the second phase of the proposed HGA method is applied. The γ best distinct solutions are then used in the second phase. This is the same post-optimisation of the proposed HGA method.

4. Experimental Results

The computational experiments are carried out to evaluate the proposed HGA. The algorithm is coded in C++ .Net 2017, and the experiments are executed on a PC with an Intel Xeon W-2133 CPU @3.60 GHz processor and 64.00 GB of RAM.

4.1. Dataset

Datasets are constructed based on the Thanet offshore wind farm that consists of 100 3MW turbines. This wind farm is located 11 km off the coast of Thanet in Kent, England. The layout of the Thanet wind farm is taken from Fischetti and Pisinger (2018). The data consists of a set of offshore turbines that need to be maintained on a given day. Table 1 presents the turbine data generated based on the dataset provided by Irawan et al. (2017). In the table, the latitude and longitude of each turbine location is given. The maintenance duration (τ_i), the weight of the equipment (w_i), and the number of technicians needed to maintain each turbine are also provided.

The initial SOV location (P_0) is located at (51.4000, 1.5600), while the final SOV location (P_f) is placed differently for each instance. The speed and fuel cost of the SOV are set to 25 km/hour and €700/hour, respectively, and those of the STB are 35 km/hour and €350/hour, respectively. The STB can accommodate up to 12 technicians on board and up to 3,900 kg of equipment and parts needed to maintain the turbines. The boarding times for crews from the SOV to the STB and from the STB to a turbine are assumed to be 5 and 11 minutes, respectively. There are at least three delivery and three pick-up trips, which are determined by the number of turbines and the crew members required. The latest recommended time ($t_{\max}$) is set to 12 hours, and a penalty cost of €700/hour is incurred if the final STB pick-up trip returns to the SOV later than $t_{\max}$. We consider Time 0 as the start time when the SOV leaves position P_0. We generate 24 instances with IxxFyy as the name of an instance. Here, Ixx refers to the number of turbines which we vary from 11 to 14. Instance I11Fyy consists of turbines T1, T2, ..., T11 whereas Instance I12Fyy includes T1, T2, ..., T12 (I13Fyy = I12Fyy $\bigcup$ {T13} and I14Fyy = I13Fyy $\bigcup$ {T14}). Fyy indicates the

Table 1: Turbine data

Turbine	Lat	Long	τ_i (hours)	w_i (kg)	ρ_i
T1	51.4455	1.6105	6	730	2
T2	51.4174	1.6380	7	359	3
T3	51.4344	1.6527	7	696	2
T4	51.4433	1.6530	7	223	3
T5	51.4093	1.6236	5	395	2
T6	51.4054	1.6417	8	561	3
T7	51.4302	1.6195	6	454	3
T8	51.4095	1.6755	7	676	2
T9	51.4512	1.6154	8	398	3
T10	51.4189	1.6097	5	674	2
T11	51.4305	1.6714	6	564	2
T12	51.4413	1.5774	5	452	3
T13	51.4271	1.6241	6	564	2
T14	51.4103	1.6614	5	452	3

fixed cost of the SOV, which we vary between €0 and €50/movement in increments of €10. For example, Instance I13F30 comprises 13 turbines with a fixed cost of €30.

4.2. Results on the discrete problem (DLMRPOWF)

To assess the performance of the proposed HGA, we compare the solutions obtained by the HGA to those obtained by the exact method (EM). Note that the MEPA described in Section 3.5 is not included in the HGA as we deal with the discrete problem. For the EM, IBM ILOG CPLEX version 12.8 is used to solve the discrete model (32)–(46) with a maximum execution CPU time of three hours (10,800 seconds). It is found that without such a time limitation, CPLEX will unfortunately terminate, as it will run out of memory. The performance of the HGA is measured using the deviation ($\%Dev$) between the Z value attained by the proposed method and the best-known solution (Z^b) obtained either from the EM or the HGA. The deviation ($\%Dev$) is formulated as follows:

$$\%Dev = \frac{Z_m - Z^b}{Z_m} \times 100 \tag{57}$$

where Z_m refers to the feasible solution cost obtained by either the EM or the HGA.

For the EM, two potential sites (Q) scenarios are implemented. First, the set of potential sites (Q) to locate SOV is restricted to the locations of the turbines (I) and the SOV's initial and final locations ($Q = I \cup \{P_0\} \cup \{P_f\}$). In the second scenario, in addition to the turbine sites, a set of midpoints between any two turbines (K) is also included ($Q = \{P_0\} \cup I \cup K \cup \{P_f\}$). Z_1^{EM} and Z_2^{EM} refer to the objective function values obtained by the EM when Scenarios 1 and 2 are used to construct the set Q, respectively. The values of the parameters used in the proposed HGA are determined based on the number of trips and turbines. Here, we set these values as follows:

the population size ($N = \min\{100, (|S| + 3)|I|\}$, the number of generations ($NG = \lfloor N/3 \rfloor$), the immigration rate ($p^e = 0.2$), and the mutation rate ($p^m = 0.1$). The HGA is executed five times for each instance to evaluate the consistency of the methods, where Z_b and Z_a refer to the best and the average objective values obtained by this method, respectively.

Table 2 presents the summary of the experimental results pertaining to the discrete problem (DLMRPOWF). As the HGA is executed five times, the table also reveals the best solution (Z_b) and the average solution (Z_a) obtained by this method. The boldface indicates the best solution found by either the HGA or the EM on the discrete model. The average deviation ($\%Dev$) and the number of best solutions found by each method are also presented.

Table 2: The experimental results on the discrete problem (DLMRPOWF)

Instance	Exact Method (3 hours)		HGA		
	Z_1^{EM}	Z_2^{EM}	Z_b	Z_a	CPU (s)
I11F0	678.33	708.48	**674.08**	681.08	27.61
I11F10	**716.76**	763.17	718.33	719.58	28.36
I11F20	749.53	766.97	**739.89**	**739.89**	27.18
I11F30	**756.76**	791.30	759.89	759.89	28.14
I11F40	778.44	**776.76**	779.89	779.89	27.76
I11F50	**796.76**	811.75	799.89	799.89	30.94
I12F0	722.26	729.67	**711.34**	717.60	43.29
I12F10	748.98	762.10	**741.34**	753.27	47.54
I12F20	765.34	794.37	**762.26**	**762.26**	48.78
I12F30	795.76	786.86	**782.26**	789.41	50.27
I12F40	802.26	842.85	**795.47**	809.68	50.06
I12F50	**805.47**	826.48	**805.47**	833.97	52.32
I13F0	814.61	790.48	**753.34**	757.93	53.48
I13F10	**778.61**	802.26	788.79	791.89	50.49
I13F20	**803.57**	856.00	815.32	815.32	50.70
I13F30	841.26	877.03	**835.32**	**835.32**	51.07
I13F40	903.91	857.14	**848.88**	854.03	55.33
I13F50	**858.13**	866.48	868.88	874.03	57.00
I14F0	807.82	848.89	**800.21**	805.61	104.13
I14F10	894.83	868.49	**834.91**	840.23	105.55
I14F20	874.23	961.24	**861.08**	863.22	109.86
I14F30	1,116.44	914.37	**881.93**	890.58	126.25
I14F40	986.19	930.03	**902.12**	905.81	129.17
I14F50	955.90	1,146.96	**921.9**9	929.69	144.97
Average % Dev	3.0111	4.7820	**0.2272**	0.8914	
# best solutions	7	1	17	3	

Based on our observation, the DLMRPOWF is difficult to solve using the EM. We notice interesting results where the EM generates worse solutions when solving second scenario problems

compared to when solving first scenario problems. The midpoints between the turbines in the set Q increase the size of the problem, making it difficult to solve optimally using the EM. The average gap between the lower bound (LB) and the upper bound (UB) obtained by CPLEX is relatively high at 16.89% and 33.63% on average for the first and the second scenarios, respectively. It is also observed that CPLEX experiences difficulties tightening the LB. According to Table 2, the proposed HGA runs well, as this method produces good solutions with an average $\%Dev$ of 0.2272% in the relatively short computational time (an average of 62.51 seconds). The average $\%Dev$ of the average solutions is not that far from the one of best solutions, demonstrating the consistency of our HGA algorithm. It is also noted that a higher SOV fixed cost tends to have a longer computational time.

4.3. Results from the continuous problem (CLMRPOWF)

As the mathematical model of the CLMRP is nonlinear, CPLEX is not able to solve the problem. Therefore, the solutions to the CLMRPOWF obtained by the proposed HGA are compared to those generated by the HPSO and HMS. Several HGA configurations are constructed by varying the λ value from 1 to 0 using intervals of 0.25. This is performed to analyse the effect of the proposed MEPA algorithm. If λ is set to 0, the MEPA is not used in the HGA method. The values of GA parameters are set to be identical to those used to solve the discrete problem, except the population size, which is set to $N = \min\{100, m \cdot |I|\}$.

Table 3 presents a summary of the experimental results pertaining to the CLMRPOWF problem, where Z_b and Z_a refer to the best and the average results of the five runs, respectively. To be consistent in our comparison, we set the computational time of all methods to that of the HGA($\lambda = 1$). According to the table, the HGA($\lambda = 1$) is found to be the best optimiser, as it produces the smallest deviation, with an average $\%Dev$ of 0.1020%. Moreover, the HGA($\lambda = 1$) obtained the best solutions for 14 instances, which is significantly more than those produced by other methods. Note that all methods utilise the VNS algorithm and the MEPA to solve the continuous problem. According to the results, the HGA with a different λ value performs better than the HPSO and the HMS. This is mainly because the genetic operators implemented in the HGA run well and produce good SOV location solutions. It is also found that decreasing the value of λ reduces the quality of the solutions produced, meaning that the MEPA algorithm significantly affects the performance of the HGA method.

A sensitivity analysis to assess the robustness of the proposed HGA method is also carried out. This is based on the injection operator and the parent selection operator. First, we remove the injection operator to evaluate whether the immigrant affects the solution. We refer to this method as the HGA-WIE. Second, the classic roulette wheel method is implemented to examine our proposed approach in the parent selection. We refer to this configuration as the HGA-RW. Note that the λ value is set to 1 for the HGA-WIE and HGA-RW.

Table 3: The experimental results on the continuous problem (CLMRPOWF)

Instance	CPU (s)	HGA ($\lambda = 1$)		HGA ($\lambda = 0.75$)		HGA ($\lambda = 0.5$)		HGA ($\lambda = 0.25$)		HGA ($\lambda = 0$)		HPSO		HMS	
		Z_b	Z_a	Z_b	Z_a	Z_b	Z_a	Z_b	Z_a	Z_b	Z_a	Z_b	Z_a	Z_b	Z_a
I11F0	26	676.40	677.76	676.39	678.60	676.40	677.79	677.45	678.97	**673.10**	681.37	676.46	681.83	677.94	683.19
I11F10	26	719.78	719.85	719.78	719.85	719.78	719.83	719.78	719.84	719.77	719.82	719.89	727.37	**718.54**	727.06
I11F20	26	**739.89**	739.89	**739.89**	739.89	**739.89**	739.89	**739.89**	739.89	**739.89**	739.89	**739.89**	748.22	766.75	765.21
I11F30	27	**759.88**	759.90	759.89	759.89	**759.88**	759.89	759.89	759.89	759.89	759.89	759.89	760.13	760.56	777.39
I11F40	32	779.89	779.89	**779.88**	779.89	**779.88**	779.92	**779.88**	779.89	779.89	779.89	779.89	780.26	783.24	790.21
I11F50	27	**796.78**	799.27	799.89	799.89	799.89	799.89	799.89	799.89	799.89	799.89	799.89	799.98	800.07	801.80
I12F0	40	**703.24**	705.44	703.88	711.42	705.11	713.24	707.26	710.69	716.97	722.77	703.48	707.63	713.52	713.69
I12F10	46	**736.12**	740.52	737.01	742.21	737.03	742.94	740.92	744.55	740.91	754.20	745.26	750.42	738.43	744.80
I12F20	57	759.02	761.39	761.64	762.95	762.27	766.14	**758.58**	765.05	766.59	777.80	761.86	776.11	768.98	768.56
I12F30	56	776.64	779.95	777.15	781.09	781.35	782.47	781.64	782.20	**776.15**	785.93	782.38	784.10	787.17	785.18
I12F40	63	**795.45**	797.62	**795.45**	798.15	**795.45**	801.52	**795.45**	803.74	**795.45**	802.23	**795.45**	796.82	798.02	795.97
I12F50	49	**805.45**	805.45	**805.45**	805.45	**805.45**	808.63	**805.45**	813.05	**805.45**	823.47	**805.45**	805.45	**805.45**	805.45
I13F0	49	**738.88**	739.36	**738.88**	739.50	738.89	740.05	739.67	744.06	739.67	747.37	739.24	741.25	740.82	742.89
I13F10	45	**775.08**	778.90	776.55	777.92	776.55	777.16	778.13	784.22	778.87	788.25	782.05	786.66	781.81	782.02
I13F20	67	**795.07**	798.56	798.80	800.46	798.87	800.55	798.87	799.65	804.09	813.07	803.29	810.44	822.93	814.59
I13F30	66	**815.08**	817.43	816.01	819.03	818.13	821.08	815.53	822.07	821.07	832.15	822.11	827.12	836.91	825.43
I13F40	78	838.71	841.39	**835.07**	838.90	838.80	843.08	838.87	846.92	854.35	854.78	838.03	843.31	835.58	837.31
I13F50	74	**855.07**	855.99	855.10	857.42	**855.07**	856.66	**855.07**	866.31	858.82	869.23	855.39	856.89	855.44	857.63
I14F0	117	780.85	785.17	779.38	785.15	780.71	797.86	780.55	793.14	788.89	797.94	778.16	782.26	**776.77**	781.47
I14F10	125	826.82	834.99	**823.95**	831.08	827.84	831.99	827.83	834.62	825.59	835.41	827.15	830.37	827.14	833.91
I14F20	130	**855.34**	859.75	857.66	859.41	857.66	862.29	857.66	863.82	858.79	866.49	857.47	865.53	859.07	861.11
I14F30	136	877.48	879.38	877.48	881.73	877.49	878.77	877.59	883.56	**877.47**	881.74	877.48	884.00	877.48	879.38
I14F40	137	897.47	899.61	897.46	899.56	897.47	898.62	897.50	897.53	897.53	898.51	**894.34**	899.81	897.63	898.46
I14F50	143	**913.75**	917.02	917.42	918.04	917.47	917.74	917.47	917.87	917.47	917.75	917.46	917.74	917.50	917.02
Average %Dev		**0.1020**	0.3953	0.1638	0.4686	0.2542	0.6316	0.2765	0.8007	0.5151	1.3314	0.3356	0.8794	0.7999	1.0349
# best solutions		**14**		7		6		6		6		4		3	

Table 4 shows the experimental results of the sensitivity analysis of the proposed HGA method. The computational time of each configuration is set to be identical to the time used in the proposed HGA presented in Table 3. Each method is executed five times for each instance. The table reveals that, based on the average %Dev, without the immigrant chromosomes, the quality of the solutions slightly deteriorates. The results also show that the proposed parent selection method performs better than the roulette wheel method. It is worthwhile to note that the HGA-WIE and HGA-RW generate more best solutions than the HGA($\lambda = 1$). However, the HGA($\lambda = 1$) consistently produces good solutions for different instances.

Table 4: Comparison results on the CLMRPOWF using different configurations used in the HGA

Instance	CPU (s)	HGA ($\lambda = 1$)		HGA-WIE		HGA-RW	
		Z_b	Z_a	Z_b	Z_a	Z_b	Z_a
I11F0	26	676.40	677.76	**676.39**	677.80	677.91	678.91
I11F10	26	**719.78**	719.85	719.79	719.85	**719.78**	719.85
I11F20	26	739.89	739.89	739.89	739.89	**739.88**	739.89
I11F30	27	**759.88**	759.90	**759.88**	759.89	**759.88**	759.89
I11F40	32	**779.89**	779.89	**779.89**	779.89	**779.89**	779.89
I11F50	27	796.78	799.27	**796.75**	798.63	799.89	799.89
I12F0	40	703.24	705.44	703.87	707.04	**702.71**	708.67
I12F10	46	736.12	740.52	736.03	741.28	**735.35**	741.12
I12F20	57	759.02	761.39	762.26	762.86	**756.74**	761.90
I12F30	56	**776.64**	779.95	781.63	782.16	777.59	782.99
I12F40	63	**795.45**	797.62	**795.45**	796.98	**795.45**	799.97
I12F50	49	**805.45**	805.45	**805.45**	805.45	**805.45**	808.84
I13F0	49	**738.88**	739.36	738.89	740.32	739.67	739.91
I13F10	45	775.08	778.90	**775.07**	778.43	**775.07**	778.59
I13F20	67	**795.07**	798.56	798.87	799.50	798.87	804.98
I13F30	66	815.08	817.43	**815.07**	817.97	818.87	825.57
I13F40	78	838.71	841.39	**835.07**	838.93	838.74	842.77
I13F50	74	**855.07**	855.99	855.08	857.41	858.87	860.27
I14F0	117	780.85	785.17	779.14	781.36	**777.55**	782.20
I14F10	125	**826.82**	834.99	**826.82**	830.02	827.68	831.98
I14F20	130	855.34	859.75	**854.38**	860.88	855.82	861.08
I14F30	136	877.48	879.38	**877.47**	880.85	877.50	883.21
I14F40	137	897.47	899.61	**897.46**	902.24	**897.46**	898.88
I14F50	143	**913.75**	917.02	917.46	918.42	917.47	918.02
Average %Dev		0.0610	0.3542	0.1136	0.3724	0.1400	0.5353
#Best Solutions		11		13		12	

We also present the best solution by solving either the DLMRPOWF or the CLMRPOWF using the solution methods used. Table 5 presents a summary of the best solution configuration where the best known total cost (Z^*), the percentage of the SOV travelling cost ($\%Z^s$), the CTV travelling cost ($\%Z^c$), and the penalty cost ($\%Z^p$) are given. The table also provides the number of

movements of the SOV (#Move) in the best solution. Information on the method that generates the best solution is also provided. Table 3 reveals that in the best known solution, the SOV travelling cost (Z^s) comprises nearly 36% of the total maintenance cost. The solution is also constructed to avoid the penalty cost ($\%Z^p = 0.00$). The table shows that, as expected, increasing the SOV fixed cost reduces the number of SOV movements, as each movement comes with additional cost. In general, when the fixed cost is set to €50, two movements occur, meaning that only one potential site ($\tilde{q}$) is selected. In other words, the SOV moves from the initial location P_0 to $\tilde{q}$, then from location $\tilde{q}$ to the final destination P_f.

Table 5: Best solution configuration of the problem

Instance	Z^*	$\%Z^s$	$\%Z^c$	$\%Z^p$	#Move	Solution methods
I11F0	673.10	41.58	58.42	0.00	5	HGA($\lambda = 0$)
I11F10	716.76	39.35	60.65	0.00	3	EM with $Q = I$
I11F20	739.88	40.82	59.18	0.00	3	HGA-RW
I11F30	756.76	42.56	57.44	0.00	2	EM with $Q = I$
I11F40	776.76	44.04	55.96	0.00	2	EM with $Q = I \cup R$
I11F50	796.75	45.44	54.56	0.00	2	HGA-WIE
I12F0	702.71	27.43	72.57	0.00	6	HGA-RW
I12F10	735.35	29.93	70.07	0.00	3	HGA-RW
I12F20	756.74	30.27	69.73	0.00	3	HGA-RW
I12F30	776.15	32.17	67.83	0.00	2	HGA($\lambda = 0$)
I12F40	795.45	28.80	71.20	0.00	1	HGA($\lambda = 0, 0.25, 0.5, 0.75, 1$),HGA-WIE,HGA-RW,PSO
I12F50	805.45	29.68	70.32	0.00	1	HGA($\lambda = 0, 0.25, 0.5, 0.75, 1$),HGA-WIE,HGA-RW,PSO,MS
I13F0	738.88	30.67	69.33	0.00	5	HGA($\lambda = 0.75, 1$)
I13F10	775.07	31.54	68.46	0.00	2	HGA-WIE, HGA-RW
I13F20	795.07	33.27	66.73	0.00	3	HGA($\lambda = 1$)
I13F30	815.07	34.90	65.10	0.00	2	HGA-WIE
I13F40	835.07	36.46	63.54	0.00	2	HGA($\lambda = 0.75$)
I13F50	855.07	37.95	62.05	0.00	2	HGA($\lambda = 0.25, 0.5, 1$)
I14F0	776.77	34.29	65.71	0.00	6	MS
I14F10	823.95	35.57	64.43	0.00	3	HGA($\lambda = 0.75$)
I14F20	854.38	37.81	62.19	0.00	3	HGA-WIE
I14F30	877.47	35.97	64.03	0.00	2	HGA($\lambda = 0$), HGA-WIE
I14F40	894.34	38.23	61.77	0.00	2	PSO
I14F50	913.75	39.70	60.30	0.00	2	HGA($\lambda = 1$)

4.4. The algorithm performance analysis

In this subsection, the analysis of the algorithm performance is presented. All the solution methods (the HGA, the HPSO, and the HMS) consist of two phases, and the same post-optimisation is applied in the second phase of all methods. Note that in the HGA, we use the results when $\lambda = 1$.

We first analyse the performance of the first phase of the solution methods first, followed by the effect of the optimisation phase.

4.4.1. The analysis of the first phase

As each instance is solved multiple times, the results generated by the first phase of each solution method are presented using the box plot in Figure 8. The figure reveals that the first phase of the HGA produces solutions that yield the smallest total cost in almost all instances. Moreover, the variance of solutions generated by the first phase of the HGA is much lower than the other two methods. This indicates that the proposed HGA is a robust tool for solving this kind of problem.

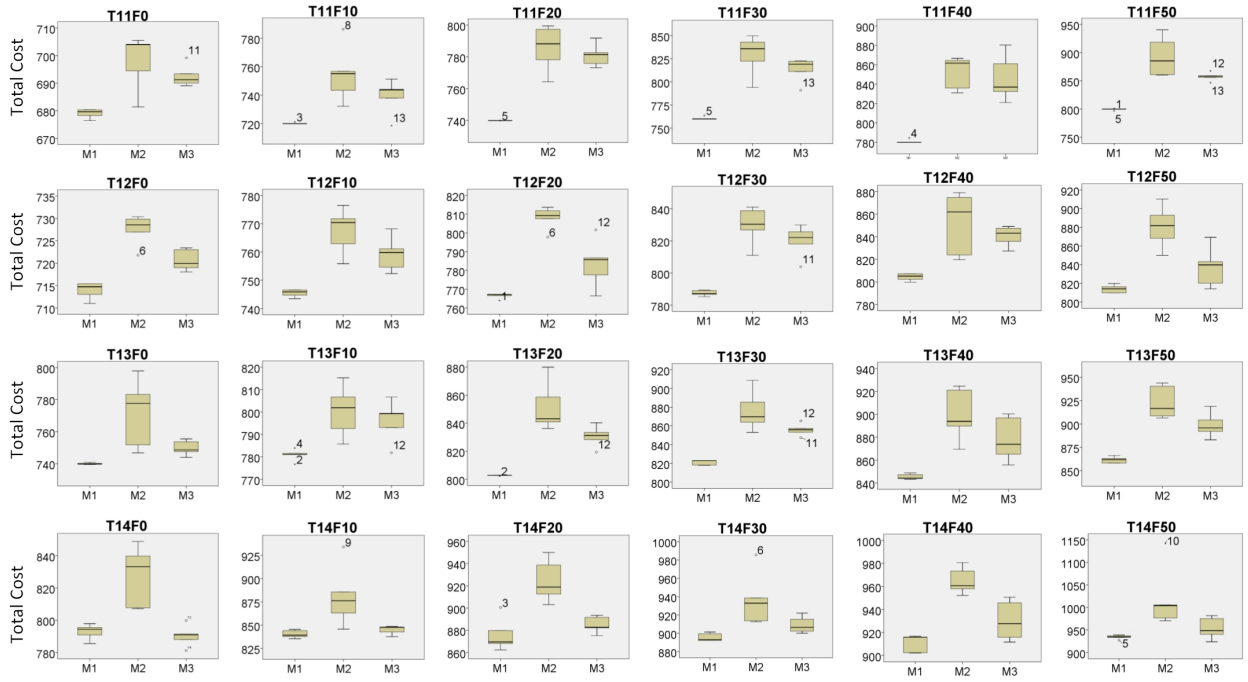

Figure 8: The box plots of the results generated by the first phase of the solution methods, where M1, M2 and M3 represent the HGA, the HPSO, and the HMS, respectively

We also perform a parametric test using the independent samples t-test to determine whether there is statistical evidence that the proposed HGA is significantly better than the other methods. We first compute the deviation for the average and best results using Equation (57) to compare the HGA vs the HPSO and the HGA vs the HMS. Figure 9 shows the $p-$values (Sig.) of the Levene's test for equality of variances and the t-test for equality of means.

Figure 9 reveals that based on both average and best results, the deviation mean and variance of the HGA are much smaller than those of the HPSO and the HMS. This means that the HGA produces better and consistent solutions compared to the other two. Based on the t-test results, we conclude that there is statistical evidence that the HGA significantly outperforms the HPSO and the HMS at 1% significance levels for both average and best results.

a) HGA vs HPSO (Deviation of best results)

Method	N	Mean	Std. Deviation	Std. Error Mean
HGA	24	0.046654	0.1369491	0.0279546
HPSO	24	3.378804	1.7739368	0.3621033

	Levene's Test for Equality of Variances		t-test for Equality of Means		
	F	Sig.	t	df	Sig. (2-tailed)
Equal variances assumed	51.331	0.000	-9.175	46	0.000
Equal variances not assumed			-9.175	23.274	0.000

b) HGA vs HMS (Deviation of best results)

Method	N	Mean	Std. Deviation	Std. Error Mean
HGA	24	0.046654	0.1369491	0.0279546
HMS	24	1.848133	1.6893621	0.3448396

	Levene's Test for Equality of Variances		t-test for Equality of Means		
	F	Sig.	t	df	Sig. (2-tailed)
Equal variances assumed	45.192	0.000	-5.207	46	0.000
Equal variances not assumed			-5.207	23.302	0.000

c) HGA vs HPSO (Deviation of average results)

Method	N	Mean	Std. Deviation	Std. Error Mean
HGA	24	0.013396	0.0656259	0.0133958
HPSO	24	5.511658	2.0765969	0.4238836

	Levene's Test for Equality of Variances		t-test for Equality of Means		
	F	Sig.	t	df	Sig. (2-tailed)
Equal variances assumed	31.325	0.000	-12.965	46	0.000
Equal variances not assumed			-12.965	23.046	0.000

d) HGA vs HMS (Deviation of average results)

Method	N	Mean	Std. Deviation	Std. Error Mean
HGA	24	0.013396	0.0656259	0.0133958
HMS	24	3.021221	2.0235057	0.4130464

	Levene's Test for Equality of Variances		t-test for Equality of Means		
	F	Sig.	t	df	Sig. (2-tailed)
Equal variances assumed	47.105	0.000	-7.278	46	0.000
Equal variances not assumed			-7.278	23.048	0.000

Figure 9: The t-test results for comparing the HGA vs the HPSO and the HGA vs the HMS

4.4.2. The effect of the post-optimisation phase

In this subsection, we examine the performance of Phase 2 (post-optimisation) in Algorithm 1. In this phase, a full VNS is implemented in which seven neighbourhood structures ($k_{\max} = 7$) are used. For consistency and simplicity, in the MEPA algorithm, we set the parameter ϵ to 0.000001. Table 6 reveals the improvement made by the post-optimisation process in each instance. The table presents the improvement in the total cost ($\%Z$) together with the additional computational time (%CPU) required to run the post-optimisation process. The observations are carried out for the proposed HGA, HPSO and HMS methods.

The second phase of the HGA reduces the average total maintenance cost by only 0.69%, at the expense of an additional 45.59% CPU time. This indicates that relatively good solutions have been produced by the first phase of the HGA. A significant improvement is achieved in the post-optimisation phase for the HPSO and the HMS: the total cost is improved by 5.7% and 3.07%, respectively. However, this gain comes with a significant increase in CPU time of 61.25% and 68.71%, respectively. In brief, we can conclude that the proposed HGA, even without the post optimisation phase, remains a better optimisation tool compared to the other two methods, the performances of which rely heavily on the second phase.

4.4.3. The effect of increasing the number of potential SOV sites on the discrete problem

It is worth noting that the optimal solution generated by solving the discrete model (DLMRPOWF) using the exact method (EM) can get closer and closer to the optimal solution for the continuous model (CLMRPOWF) when the number of potential sites (Q) for the SOV gets larger and larger. In other words, we can theoretically approximate a continuous problem using its counterpart, a discrete problem, if an infinite number of potential sites is adopted. However, in practice a larger $|Q|$ makes the problem more difficult to solve by CPLEX and hence the optimal solution

Table 6: Improvement by the post-optimisation phase

Instance	HGA		HPSO		HMS	
	$\%Z$	%CPU	$\%Z$	%CPU	$\%Z$	%CPU
I11F0	0.19	54.65	2.30	57.94	1.36	74.04
I11F10	0.04	45.22	3.63	54.86	1.64	62.50
I11F20	0.00	32.99	4.74	48.22	2.02	66.28
I11F30	0.10	41.61	8.31	70.51	4.40	72.72
I11F40	0.12	87.13	8.41	175.20	6.64	236.54
I11F50	0.02	58.93	10.42	144.55	6.49	189.43
I12F0	1.19	56.95	2.73	63.63	0.97	63.79
I12F10	0.66	47.90	2.22	50.48	1.90	50.47
I12F20	0.66	31.21	3.95	37.47	1.92	40.16
I12F30	0.96	31.79	5.48	47.72	4.23	53.73
I12F40	0.83	54.73	6.46	70.70	5.31	83.81
I12F50	1.05	44.03	8.53	73.63	3.80	76.50
I13F0	0.09	72.43	3.92	70.91	0.93	69.78
I13F10	0.25	56.48	1.72	64.73	1.76	69.67
I13F20	0.52	30.50	4.85	38.87	1.92	39.04
I13F30	0.44	43.99	5.60	62.57	3.56	63.57
I13F40	0.49	69.21	6.27	83.52	4.67	80.53
I13F50	0.65	78.83	7.19	80.10	4.58	79.22
I14F0	0.97	24.06	5.45	26.42	1.12	27.07
I14F10	0.64	23.30	5.74	23.54	1.30	26.49
I14F20	1.86	22.52	6.39	25.12	2.70	26.79
I14F30	1.81	27.21	5.62	29.13	3.29	30.58
I14F40	1.20	27.99	6.75	38.69	3.42	33.36
I14F50	1.85	30.60	10.01	31.57	3.87	33.09
Average	0.69	45.59	5.70	61.25	3.07	68.71

will not be guaranteed if the limited computational time allocated is not long enough.

Here, we perform experiments where we can guarantee optimality by using a small instance consisting of six turbines. To see whether a similar pattern may exist under different fixed costs, we opted for two levels of fixed costs, namely, a fixed cost of 0 and 20 respectively. In addition to the turbine sites (I), the set Q is augmented by including additional potential sites (25, 64, 100, 225 and 400) using a rectangle grid covering the turbine locations. The EM is then solved using CPLEX for each of these discrete problems. For simplicity, six potential sites (Q) scenarios with an increasing number of potential sites from one scenario to the next are implemented: (i) $(EM-I)$: $Q = I \cup \{P_0\} \cup \{P_f\}$, (ii) $(EM-G25)$: $(Q = \{P_0\} \cup I \cup G^{25} \cup \{P_f\})$, (iii) $(EM-G64)$: $(Q = \{P_0\} \cup I \cup G^{64} \cup \{P_f\})$, (iv) $(EM-G100)$: $(Q = \{P_0\} \cup I \cup G^{100} \cup \{P_f\})$, (v) $(EM-G225)$: $(Q = \{P_0\} \cup I \cup G^{225} \cup \{P_f\})$, and (vi) $(EM-G400)$: $(Q = \{P_0\} \cup I \cup G^{400} \cup \{P_f\})$, Here, $G25$, $G64$, $G100$, $G225$ and $G400$ represent 25, 64, 100, 225 and 400 extra potential sites constructed in the grid, respectively where the new potential sites are the intersection points of the x and y axis of the rectangles inside the grid. For example, the 25 additional points are the intersection points of the 5 rows $\times$ the 5 columns of the grid. Note that the solution of the continuous problem is

solved using the HGA($\lambda = 1$) with the set Q is constructed based on turbines sites and midpoints between the turbines only. For each discrete problem, the deviation (in %) from the solution of the continuous problem is computed and the corresponding CPU time (in secs) is recorded.

Figure 10 shows the solutions generated by the EM over the number of points (up to 400 points). For Instance in the case of T6F0 in Figure 10(a), the cost reduces or does not increase as the number of potential sites increases while the corresponding CPU times increases drastically. However, it was also observed that for the case of T6F20 in Figure 10(b), the pattern was not followed as expected where the solution of EM-$G225$ was worse. One may presume that the result contradicts the theoretical claim given the instance with a relatively smaller number of additional points such as the EM-$G64$ and the EM-$G100$ obtain better results. However, this is not unusual as these smaller instances must have included more promising potential sites than those used for EM-$G225$ given the was these points are generated. This observation needs to be taken into account when adopting such an approximation type strategy. As mentioned earlier, including a very large number of potential points or a large number of promising points will normally converge to the optimal solution of the continuous problem.

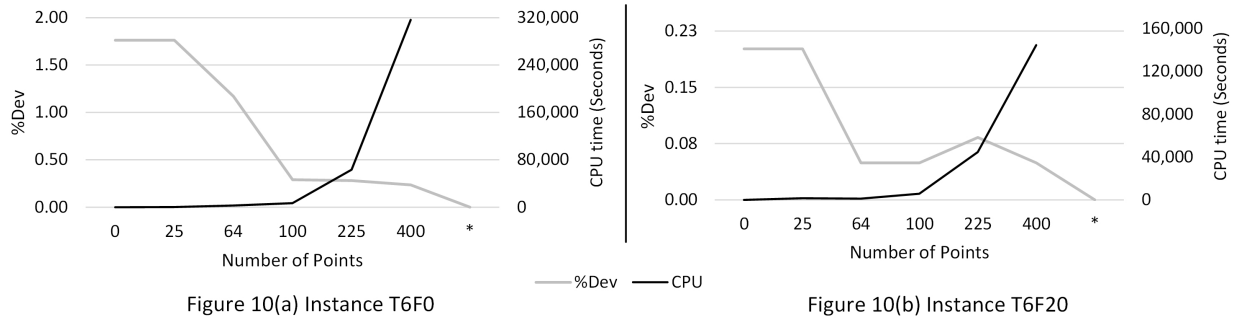

Figure 10: The solutions generated by the EM over the number of points

To evaluate the performance of the proposed HGA, we solve the same instance based on the continuous model (CLMRPOWF) as performed previously using all the various fixed costs but using up to 100 extra potential sites only. This is due to the large CPU time that is required to solve the discrete problem as reported in Figure 10. Table 7 presents the results of the small instance generated by the EM and the HGA, where the total cost (Z) together with the average %Dev are provided. The CPU time is measured in seconds. The HGA is executed five times to obtain the average and best costs. Table 7 also shows that the proposed HGA runs well when solving continuous problems. All solutions produced by the HGA are still better than those optimal solutions produced for the discrete problem by the EM. It also means that more promising points are required to improve the quality of solutions generated by the EM on the discrete problem. This could increase the chance to attain good quality solutions at the expense of a longer computational time as shown by the average %Dev when 64 and 100 points in the grid are used.

Table 7: Results comparison on the small instance with large number of potential sites

Instance	EM−I		EM−$G25$		EM−$G64$		EM−$G100$		HGA		
	Z	CPU	Z	CPU	Z	CPU	Z	CPU	Z_b	Z_a	CPU
T6F0	553.96	4	553.96	106	550.75	2,754	545.97	6,872	544.38	544.46	10
T6F10	583.76	5	583.76	479	582.85	2,900	582.85	5,350	582.48	583.11	7
T6F20	603.76	5	603.76	1,614	602.85	1,242	602.85	5,682	602.55	603.07	9
T6F30	623.76	4	623.76	1,454	622.85	1,681	622.85	4,177	622.49	622.82	9
T6F40	643.76	2	643.76	75	642.85	1,656	642.85	9,030	642.59	642.75	9
T6F50	663.76	4	663.76	1,526	662.85	1,765	662.85	3,586	662.52	662.90	9
Average	0.459%	4	0.459%	876	0.238%	2,000	0.091%	5,783	**0.000%**	0.057%	9

4.4.4. The comparison with other known algorithms on a related problem

Our proposed method is also adapted to solve the related classic problem addressed in the literature: the planar single-facility location routing problem (PSFLRP). Here, parts of the proposed algorithms are used, including the VNS algorithm and the MEPA. In the PSFLRP, the delivery and pick-up trips are not considered. Moreover, the visiting time at each node is not determined. Here, we slightly modify our algorithm to be able to solve the problem. The results of our method are compared to those obtained by Schwardt and Dethloff (2005), Schwardt and Fischer (2009), Salhi and Nagy (2009), and Manzour-al-Ajdad et al. (2012). Note that Schwardt and Fischer (2009) propose two methods: the sequential method (Schwardt and Fischer[1]) and the neural network technique (Schwardt and Fischer[2]). Seven instances taken from Manzour-al-Ajdad et al. (2012) are solved, and Table 8 shows the results on the PSFLRP. According to the results, our method outperforms other algorithms, as it produces the smallest average %Dev of 0.2634%, in addition to discovering five new best solutions. This indicates that our approach is robust and flexible enough to solve related LRPs.

Table 8: Results comparison with other algorithms in solving the planar single-facility location routing problem

Instance	Schwardt and Dethloff	Schwardt and Fischer[1]	Schwardt and Fischer[2]	Salhi and Nagy	Manzour-al-Ajdad et al.	HGA		
						Best	Avg	CPU(s)
C1 - 50	**521.4**	527.7	522.5	522.8	538.5	**521.4**	529.7	9
C2 - 75	888.8	881.9	879.0	854.1	849.8	**834.4**	845.8	79
C3 - 100	837.7	864.5	835.6	838.5	858.8	**826.9**	827.8	103
C4 - 100	827.0	826.0	828.9	830.4	821.6	**818.7**	846.0	110
C6 - 120	907.4	931.8	905.7	972.5	902.7	**897.4**	900.3	279
C8 - 150	1,068.8	1,110.9	1,058.4	1,091.2	1,088.2	**1,028.7**	1,037.6	374
C10 - 199	1,363.5	1,377.1	**1,344.0**	1,373.2	1,354.8	1,369.2	1,378.5	1,198
Avg Dev(%)	2.1019	3.6176	1.6135	2.9941	2.2747	**0.2634**	1.4210	

4.5. Managerial Insight

In this subsection, we provide a scenario analysis to highlight (i) the effect of the fixed cost on the number of SOV stops, and (ii) the impact the running (travel) cost of the two vessels may have on the total cost and hence on the viability of the solution configuration.

(i) Effect of the fixed cost

If we refer back to Table 5, it is worth noting that in the best-known solution, the SOV traveling cost (Z^s) comprises almost 36% of the total maintenance cost. The solution is also constructed to avoid the penalty cost. In this experiment, we generate the solution by varying the fixed cost from 0 to 50 in increments of 10. In each solution configuration we also record the number of SOV stops. For example, Figure 11a provides the number of SOV stops in the best solution for Instance I12. A similar pattern is observed for the other instances. Note that the number of stops includes the stop at the final destination (P_f). As expected, increasing the SOV fixed cost reduces the number of stops to the point where the solution configuration stabilises at two stops. This observation may not be valid if the fixed cost is relatively high, as the additional cost for each movement (stop and start) could become too expensive, resulting in one or no stops. For example, for I12, the fixed cost of €40 leads to only one stop, whereas for I14, the solution stabilises at two stops when the fixed cost reaches €30.

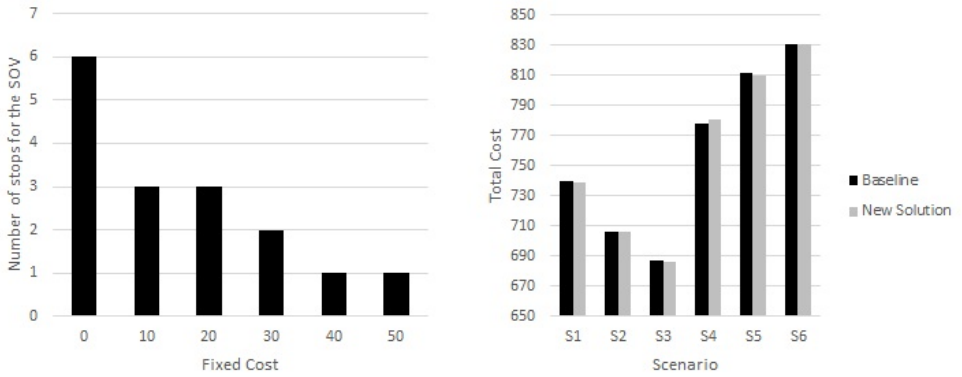

Figure 11: The scenario analysis for Instance I12

(ii) Effect of variable cost on the robustness of the solution

We examine the effect that the travel cost of both the SOV and the STB has on the robustness of the solution configuration. In other words, we want to explore whether small changes in the running cost have significant effects on the solution configuration (and hence the schedule); any significant changes may inconvenience. We refer to the original configuration (or schedule) as the base line, the total cost of which is computed by substituting the new running cost in the solution and comparing it to that in the new solution that is found when the changes in running cost are introduced. In the previous experiments, the travel costs of the SOV and the STB are set to

€700/hour and €350/hour, respectively. We construct six scenarios by increasing or decreasing the travel cost by 10%. For instance, In Scenario 1 (S1), the travel cost of the SOV is reduced by 10% to €630/hour, while the running cost of the STB remains unchanged at €350/hour; In Scenario 2 (S2), the STB travel cost is reduced by 10% to €315/hour, while the SOV remains unchanged. The running cost of the remaining four scenarios are summarised as follows: Scenario 3 (S3): SOV (€630/hour) and STB (€315/hour); Scenario 4 (S4): SOV (€770/hour) and STB (€350/hour); Scenario 5 (S5): SOV (€700/hour) and STB (€385/hour); Scenario 6 (S6): SOV (€770/hour) and STB (€385/hour).

To illustrate such scenarios, we consider the problem in Instance I12, which has a fixed cost of €20, as similar trends are also observed in the other instances. Figure 11b presents the total maintenance cost for each scenario. Here, two solutions are used: the baseline and new solutions. As mentioned earlier, the baseline solution is obtained by solving the problem using the HGA when the travel costs of the SOV and STB are set to their original values of €700/hour and €350/hour, respectively. This solution configuration is unchanged but its total cost is revaluated for each scenario. The new solution for each scenario is determined by implementing the HGA to solve each problem scenario using the respective travel costs. The figure reveals that both solutions produce similar total costs with a miniscule deviation of 0.0317%. This demonstrates that the proposed HGA produces solution configurations (or schedules) that are robust to the changes due to the running costs. This observation is useful to decision makers given the considerable fluctuations in the fuel costs.

5. Conclusion and Suggestions

In this paper, a new and challenging green logistic problem that integrates continuous location and maintenance routing problem for offshore wind farms was investigated. The aim was to dynamically identify the best SOV locations in the plane and the best delivery and pick-up routes for the STB to use when maintaining the offshore turbines. We first developed optimisation models based on MINLP for the CLMRPOWF and for its counterpart, the DLMRPOWF.

We also designed an efficient HGA by integrating a GA and VNS. To achieve near-optimal solutions for the continuous locations, we introduced a modification of the end-point algorithm (MEPA) initially designed for the planar LRP based on the well-known Weiszfeld iterative process. We assessed the effectiveness of the proposed HGA by comparing the results to those of a new HMS heuristic and an efficient evolutionary approach (PSO). Computational experiments were conducted using the Thanet wind farm in the UK as an example. The results demonstrate that the proposed hybrid evolutionary method that integrates the GA with VNS and other elements from continuous location analysis is an effective tool for solving the CLMRPOWF. Managerial insights that cover the effect of the fixed cost on the number of stops, and whether changes in the travel cost of the two vessels have a significant effect on the solution configuration and hence the schedule are provided. Our method is also found to be easily adaptable for solving a related continuous LRP (PSFLRP),

as it outperformed the state-of-the-art methods available in the literature.

A number of aspects could be worth exploring further. The study could, for example, be built upon to include multiple periods, with the last location of a given period acting as the starting location for the next period. Another avenue would be to extend the model by considering the effect of uncertainty on some of the parameters, such as the maintenance duration together with travel and transfer time. The way we augmented the discrete problem with artificial potential sites is based on a simple grid type approach but this could be revisited by constructing a more effective mechanism to generate promising additional potential sites instead. Such a scheme will enhance the solution quality while requiring a relatively less computational burden.

In this study, we encounter no geographical obstacles, but that might not be case in other areas, including in which travel time and location are restricted. From a practical viewpoint, the above techniques could easily be used as a basis for the design of an efficient decision-support system for offshore wind farms to be used by environmental agencies and governments that embrace the green economy.

References

Almouhanna, A., Quintero-Araujo, C.L., Panadero, J., Juan, A.A., Khosravi, B., Ouelhadj, D., 2020. The location routing problem using electric vehicles with constrained distance. Computers & Operations Research 115, 104864.

Bettinelli, A., Cacchiani, V., Crainic, T.G., Vigo, D., 2019. A branch-and-cut-and-price algorithm for the multi-trip separate pickup and delivery problem with time windows at customers and facilities. European Journal of Operational Research 279, 824–839.

Brimberg, J., Drezner, Z., Mladenović, N., Salhi, S., 2014. A new local search for continuous location problems. European Journal of Operational Research 232, 256–265.

Calık, H., Oulamara, A., Prodhon, C., Salhi, S., 2021. The electric location-routing problem with heterogeneous fleet: Formulation and benders decomposition approach. Computers & Operations Research 131, 105251.

Cao, J.X., Zhang, Z., Zhou, Y., 2021. A location-routing problem for biomass supply chains. Computers & Industrial Engineering 152, 107017.

Cooper, L., 1964. Heuristic methods for location-allocation problems. SIAM Review 6, 37–53.

Dai, L., Stålhane, M., Utne, I.B., 2015. Routing and scheduling of maintenance fleet for offshore wind farms. Wind Engineering 39, 15–30.

Dantzig, G.B., Wolfe, P., 1960. Decomposition principle for linear programs. Operations Research 8, 101–111.

Dewan, A., Asgarpour, M., 2016. Reference O&M Concepts for Near and Far Offshore Wind Farms. Technical Report. ECN-E-16-055.

Fischetti, M., Pisinger, D., 2018. Optimizing wind farm cable routing considering power losses. European Journal of Operational Research 270, 917 – 930.

Ghaderi, A., Jabalameli, M., Barzinpour, F., Rahmaniani, R., 2012. An Efficient Hybrid Particle Swarm Optimization Algorithm for Solving the Uncapacitated Continuous Location-Allocation Problem. Networks and Spatial Economics 12, 421–439.

Ghaffarinasab, N., Van Woensel, T., Minner, S., 2018. A continuous approximation approach to the planar hub location-routing problem: Modeling and solution algorithms. Computers & Operations Research 100, 140–154.

Gillett, B., Miller, L., 1974. A heuristic algorithm for the vehicle dispatch problem. Operations Research 22, 340–344.

Hansen, P., Mladenović, N., Pérez, J.M., 2010. Variable neighbourhood search: methods and applications. Annals of Operations Research 175, 367–407.

Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S., 2017. Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization 5, 423–454.

Holland, J.H., 1992. Genetic algorithms. Scientific American 267, 66–73.

Irawan, C.A., Eskandarpour, M., Ouelhadj, D., Jones, D., 2021. Simulation-based optimisation for stochastic maintenance routing in an offshore wind farm. European Journal of Operational Research 289, 912–926.

Irawan, C.A., Ouelhadj, D., Jones, D., Stålhane, M., Sperstad, I.B., 2017. Optimisation of maintenance routing and scheduling for offshore wind farms. European Journal of Operational Research 256, 76 – 89.

Karakostas, P., Sifaleras, A., Georgiadis, M.C., 2020. Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. Expert Systems with Applications 153, 113444.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, pp. 1942–1948 vol.4.

Lin, S., 1965. Computers solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269.

Lopes, R.B., Ferreira, C., Santos, B.S., Barreto, S., 2013. A taxonomical analysis, current methods and objectives on location-routing problems. International Transactions in Operational Research 20, 795–822.

Manzour-al-Ajdad, S., Torabi, S., Salhi, S., 2012. A hierarchical algorithm for the planar single-facility location routing problem. Computers & Operations Research 39, 461–470.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24, 1097–1100.

Nagy, G., Salhi, S., 2007. Location-routing: Issues, models and methods. European Journal of Operational Research 177, 649 – 672.

Poikonen, S., Golden, B., 2020a. The mothership and drone routing problem. INFORMS Journal on Computing 32, 249–262.

Poikonen, S., Golden, B., 2020b. Multi-visit drone routing problem. Computers & Operations Research 113, 104802.

Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. European Journal of Operational Research 238, 1 – 17.

Rabbani, M., Heidari, R., Yazdanparast, R., 2019. A stochastic multi-period industrial hazardous waste location-routing problem: Integrating nsga-ii and monte carlo simulation. European Journal of Operational Research 272, 945 – 961.

Rabie, H.M., El-khodary, I.A., Tharwat, A.A., 2013. A particle swarm optimization algorithm for the continuous absolute p-center location problem with euclidean distance. International Journal of Advanced Computer Science and Applications 4, 101–106.

Raknes, N.T., Ødeskaug, K., Stålhane, M., Hvattum, L.M., 2017. Scheduling of maintenance tasks and routing of a joint vessel fleet for multiple offshore wind farms. Journal of Marine Science and Engineering 5, 1.

Redondo, J.L., Fernández, J., García, I., Ortigosa, P.M., 2009. Solving the multiple competitive facilities location and design problem on the plane. Evolutionary computation 17, 21–53.

Roberti, R., Ruthmair, M., 2021. Exact methods for the traveling salesman problem with drone. Transportation Science 55, 315–335.

Rybičková, A., Mocková, D., Teichmann, D., 2019. Genetic algorithm for the continuous location-routing problem. Neural Network World 29, 173–187.

Salhi, S., 1987. The Integration of Routing into the Location-Allocation and Vehicle Fleet Composition Problems. Technical Report. PhD dissertation, Lancaster University.

Salhi, S., 2017. Heuristic Search: The Emerging Science of Problem Solving. Springer, Switzerland.

Salhi, S., Gamal, M., 2003. A genetic algorithm based approach for the uncapacitated continuous location-allocation problem. Annals of Operations Research 123, 203–222.

Salhi, S., Nagy, G., 2009. Local improvement in planar facility location using vehicle routing. Annals of Operations Research 167, 287–296.

Salhi, S., Petch, R.J., 2007. A ga based heuristic for the vehicle routing problem with multiple trips. Journal of Mathematical Modelling and Algorithms 6, 591–613.

Salhi, S., Rand, G.K., 1989. The effect of ignoring routes when locating depots. European Journal of Operational Research 39, 150 – 156.

Salhi, S., Sari, M., 1997. A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. European Journal of Operational Research 103, 95 – 112.

Schneider, M., Drexl, M., 2017. A survey of the standard location-routing problem. Annals of Operations Research 259, 389–414.

Schrotenboer, A.H., uit het Broek, M.A., Jargalsaikhan, B., Roodbergen, K.J., 2018. Coordinating technician allocation and maintenance routing for offshore wind farms. Computers & Operations Research 98, 185 – 197.

Schrotenboer, A.H., Ursavas, E., Vis, I.F., 2020. Mixed integer programming models for planning maintenance at offshore wind farms under uncertainty. Transportation Research Part C: Emerging Technologies 112, 180 – 202.

Schrotenboer, A.H., Ursavas, E., Vis, I.F.A., 2019. A branch-and-price-and-cut algorithm for resource-constrained pickup and delivery problems. Transportation Science 53, 1001–1022.

Schwardt, M., Dethloff, J., 2005. Solving a continuous location-routing problem by use of a self-organizing map. International Journal of Physical Distribution & Logistics Management 35, 390–408.

Schwardt, M., Fischer, K., 2009. Combined location-routing problems - a neural network approach. Annals of Operations Research 167, 253–269.

Snyder, B., Kaiser, M.J., 2009. Ecological and economic cost-benefit analysis of offshore wind energy. Renewable Energy 34, 1567 – 1578.

Stålhane, M., Hvattum, L.M., Skaar, V., 2015. Optimization of routing and scheduling of vessels to perform maintenance at offshore wind farms. Energy Procedia 80, 92 – 99.

Stock-Williams, C., Swamy, S.K., 2019. Automated daily maintenance planning for offshore wind farms. Renewable Energy 133, 1393–1403.

Tayebi Araghi, M.E., Tavakkoli-Moghaddam, R., Jolai, F., Hadji Molana, S.M., 2021. A green multi-facilities open location-routing problem with planar facility locations and uncertain customer. Journal of Cleaner Production 282, 124343.

Weiszfeld, E., 1937. Sur le point pour lequel la somme des distances de n points donnes est minimum. Tohoku Mathematical Journal 43, 355–386.

Yang, S., 2008. Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evolutionary Computation 16, 385–416.