Cluster Computing (2024) 27:8865–8887
https://doi.org/10.1007/s10586-024-04416-4

Hybrid metaheuristic schemes with different configurations and feedback mechanisms for optimal clustering applications

Daisy Nkele Molokomme[1] · Adeiza James Onumanyi[2] · Adnan M. Abu-Mahfouz[1,2]

Received: 6 December 2023 / Revised: 20 February 2024 / Accepted: 3 March 2024 / Published online: 12 April 2024

Abstract
This paper addresses the critical gap in the understanding of the effects of various configurations and feedback mechanisms on the performance of hybrid metaheuristics (HMs) in unsupervised clustering applications. Despite the widespread use of HMs due to their ability to leverage multiple optimization methods, the lack of comprehensive studies on their configuration and feedback mechanisms effects often results in sub-optimal clustering performances and premature convergence. To tackle these issues, we introduce two algorithms for implementing eight distinct HM schemes, focusing on the impacts of parallel and serial processing models along with different feedback mechanisms. Our approach involves selecting candidate metaheuristics based on a mix of evolutionary and swarm-based methods, including the k-means algorithm, to form various HM-based clustering schemes. These schemes were then rigorously evaluated across a range of datasets and feedback mechanisms, further assessing their efficiency in the deployment of smart grid base stations. Performance analysis was based on total fitness evaluations, timing capabilities, and clustering accuracy. The results revealed that parallel HMs with decoupled feedback mechanisms performed best in terms of accuracy but at the cost of slower convergence rates as compared to serial HMs. Our findings further suggest that serial HMs will be best suited for time-sensitive applications where a compromise between speed and accuracy is acceptable, while parallel HMs with decoupled feedback mechanisms are preferable for scenarios where precision is paramount. This research significantly contributes to the field by providing a detailed analysis of HM performance in varying conditions, thereby guiding the selection of appropriate HM schemes for specific clustering tasks.

Keywords Data clustering · Edge computing · Ensemble clustering · Localization · Metaheuristics

1 Introduction

Clustering plays a significant role in many application areas, particularly domains that require the identification of data groupings without the need for prior training, such as in smart grids [1], pattern recognition [2], and wireless communications [3], to name a few. Among the available clustering algorithms in the literature, metaheuristics are gaining prominence due to their capacity to approximate global optimal clustering solutions. They are considered for their adaptability and robustness in exploring complex search spaces, a quality highly valuable in clustering, where conventional optimization methods like k-means may underperform [4]. Furthermore, metaheuristics excel in efficiently finding satisfactory solutions for NP-hard problems, such as clustering, without relying on gradient information [5]. However, their stochastic nature can lead to inconsistencies, potentially converging to local optima

Adeiza James Onumanyi and Adnan M. Abu-Mahfouz have contributed equally to this work.

✉ Daisy Nkele Molokomme
dmolokomme@csir.co.za

Adeiza James Onumanyi
aonumanyi@csir.co.za

Adnan M. Abu-Mahfouz
a.abumahfouz@ieee.org

1 Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Hatfield, Pretoria 0028, Gauteng, South Africa

2 Next Generation Enterprises and Institutions, Council for Scientific and Industrial Research (CSIR), Meiring Naude, Pretoria 0001, Gauteng, South Africa

or necessitating extensive computational time [6]. To address these limitations, researchers are increasingly turning to hybrid metaheuristics (HMs), which combine multiple metaheuristics to strike a balance between exploration and exploitation. This approach enhances consistency and reduces the risk of premature convergence.

Concerning the use of HMs in clustering, it is widely recognized that the selection of the structural configuration strategy (i.e., how constituent metaheuristics are combined) is problem-dependent and can be influenced by various factors, including the characteristics of the optimization problem, algorithm compatibility, exploration versus exploitation requirements, and the dynamics of the search space [4]. For instance, in the context of an optimization problem in a cloud computing environment, a parallel execution strategy was advocated [4, 7]. Their decision in this regard was guided by the presence of multiple computing services typical of cloud computing environments. In contrast, in other studies [8, 9], authors favored a sequential strategy when addressing similar clustering problems. However, insufficient attention to the ramifications of the chosen structural configuration and feedback mechanisms leaves a degree of uncertainty regarding when and how these different HMs strategies should be deployed. Consequently, this knowledge gap constrains the effective combination, configuration, and feedback strategies essential for HMs to strike a balance between convergence and diversity while also improving their processing efficiency.

Consequently, there is a significant need to investigate the combined effects of various constituent algorithms and their feedback mechanisms on the performance of HMs. This is important for a deeper comprehension of their application, particularly in enhancing clustering performance. Thus, the motivation for the present research stems from the challenges faced in unsupervised clustering applications, particularly those utilizing HMs. Despite the popularity of HMs for their ability to leverage the strengths of multiple optimization methods, there exists a significant knowledge gap regarding the optimal configurations and feedback mechanisms for these approaches. This gap often results in issues like premature convergence and sub-optimal clustering outcomes, hindering the effectiveness of HMs in practical applications. Consequently, this study aims to address these challenges by exploring the combined effects of various HM configurations and feedback mechanisms, with a goal to enhance the performance of HMs in application areas such as smart grid base station optimization. Hence, in conducting this research, we make the following contributions:

1. We propose two innovative algorithms designed to operationalize eight distinct HM schemes, with a detailed examination of how serial and parallel processing models, alongside varied feedback mechanisms, influence their efficiency. Our findings demonstrate that parallel HMs equipped with decoupled feedback mechanisms outperform their serial counterparts, albeit with a trade-off in convergence speed. This marks an advancement in the understanding of HM-based clustering applications, setting a new benchmark for future research and practical applications.
2. We demonstrate that the proposed HMs match the computational efficiency of single metaheuristics by using the total fitness evaluations as the performance metrics. This critical insight challenges the existing assumptions that HMs consume more physical computation time than their serial counterparts, while underscoring the viability of HMs as a potent tool for clustering applications.
3. We propose the network coverage indicator as a measure for assessing and optimising the base station deployment problem in smart grid networks using the proposed HMs. By employing the network coverage indicator, a straightforward yet powerful metric that quantifies the connectivity performance within smart grids is introduced. Furthermore, we not only validated the algorithms' effectiveness but also showcased their potential to significantly enhance the operational efficiency of smart grid communications.

The remainder of this paper is organized as follows. Section 2 provides a concise overview of the literature on hybrid schemes for clustering purposes. In Sect. 3, the entire methodology adopted towards the development of the different HM schemes are is discussed and in Sect. 4, we present the application area regarding edge radio placement in smart grid networks. Section 5 presents and discusses the obtained results and we conclude in Sect. 6.

2 Related work

The use of HMs for clustering has gained considerable traction in the literature. These hybrid methods aim to integrate various clustering processes - either from different algorithms or from the same algorithm with varied approaches [10, 11]. This differs from traditional approaches that employ a single algorithm, often resulting in sub-optimal clustering outcomes and limited diversity. Given the large number of metaheuristics in the literature, it is almost impossible to discuss them all, and thus it is essential to adopt a systematic approach to select the candidate algorithms for a hybrid model. Consequently, minimizing computational complexity, ensuring inter-algorithm compatibility within a hybrid framework, and achieving superior solution quality and robustness remains a major goal in the literature. Furthermore, in building HMs, there are several proposals in the literature worthy of mention, which consider the execution of multiple metaheuristics that cooperate to explore the search space by exchanging solutions.

On the one hand, there is the serial hybrid model where algorithms are executed sequentially, with each subsequent algorithm refining the solution generated by its predecessor. Thus, the quality of the initial solution exerts a considerable influence on the overall optimization process, as it can shape the trajectory of subsequent search efforts. However, should an inferior solution be initially selected, it could adversely affect the search direction. This model has been investigated in many studies such as in [8, 9, 12, 13]. Commonly, these works employ the k-means algorithm as the starting point for subsequent metaheuristics such as the bee colony optimization (BCO), firefly algorithm (FFA), and particle swarm optimization (PSO). The primary rationale for choosing k-means is for its computational efficiency and near-optimality capability, which are desirable qualities that can be further refined to achieve improved outcomes. Despite the benefits exhibited by this approach, there are still some limitations that may limit its applicability to large-scale problems. One of these limitations includes its inability to explore diverse regions of the solution space simultaneously, which may lead to sub-optimal solutions. To address this challenge, an attempt to integrate the simulated annealing component into their hybrid scheme termed as SA-PSO-GK++ was presented, where gaussian estimation distribution (GED) was employed to refine the solutions achieved by PSO and K-Means++ [13].

In contrast to sequential execution, the parallel configuration has been explored as well, which forms an ensemble structure, allowing multiple algorithms to operate independently before aggregating their results through a combiner. Such approaches are well-documented in existing literature [14–17]. These parallel hybrid schemes can be sub-divided into either coupled or decoupled feedback mechanisms. In coupled mechanisms, each algorithm's results undergo a competitive evaluation phase, where optimal outcomes are recycled into a shared loop for future iterations across algorithms. Conversely, decoupled mechanisms allocate distinct loops to each algorithm, independently cycling their solutions through subsequent iterations until convergence. In the study by [14], two equally sized populations operated independently before their optimal solutions were juxtaposed in a competitive framework, aiming to discern winners and losers based on fitness values. Although parallel hybrid methods strive to cultivate solution diversity among hybrid components, they may not fully leverage the strengths of disparate search algorithms, as indicated by [18], due to inconsistencies in convergence rates and diversity preservation. In [6] a Cauchy mutation operator was employed in a HM framework based on ant lion optimization (ALO) and ant colony optimization (ACO) to escape the local optima when solving the clustering problem. Their main goal was to minimize the intra-cluster distance. Similarly a parallel execution was attempted in [19] where the feature selection method was integrated into the their proposed HM framework to better the performance of the model on a high-dimensional data.

The literature review highlights the effectiveness of diverse hybrid schemes in addressing various clustering problems. Nevertheless, their performance tends to diminish when applied to extensive and densely clustered datasets. Despite the thorough exploration in this domain, uncertainties linger concerning the specific effect that the deployment configuration and the feedback mechanisms adopted within these HM frameworks may have on clustering performance. These uncertainties regarding where and how these HM frameworks should be deployed, poses a challenge within the current body of research. Consequently, the need for algorithms to implement such studies served as motivation for the research conducted in the present paper with an intention to guide where and how HM schemes should be deployed.

3 Methodology

In this section, we describe the methodology used to develop and assess the two algorithms proposed for implementing the serial and parallel HM schemes, incorporating both coupled and decoupled feedback mechanisms. Firstly, we describe the objective function used by the different clustering algorithms for assessing data clustering quality. Then, we describe the chosen constituent algorithms for the HM schemes, following a selection process and analysis of the different potential metaheuristics. Subsequently, we describe the two algorithms for implementing the hybrid configurations under consideration and highlight their different time complexities and associated termination criteria.

3.1 Objective function

In clustering applications, an objective (or fitness) function is used to evaluate how well data points are grouped into clusters, hence, serving as a measure of the quality of solutions obtained in the clustering problem. In this paper, we employed the within (or intra) cluster distance, W, as the objective function in each algorithm because of its well-known use and effectiveness in the literature.

Technically, the within cluster distance aims to minimize the sum distances between the data points and their respective cluster centroids, which makes it effective for clustering purposes.

Specifically, we computed W as the sum of the Euclidean distances between each data point and the centroid of their associated cluster. Mathematically, it can be described as:

$$W = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\| \tag{1}$$

where k is the number of clusters, C_i is the i-th cluster, x is a data point in cluster C_i, μ_i is the centroid of cluster C_i, and $\|\cdot\|$ denotes the Euclidean distance. In metaheuristics, the objective function W guides the search process towards optimal or near-optimal cluster assignments. A lower value of W indicates better clustering, where data points are closer to their respective centroids, thus forming more cohesive clusters. Thus, in order to evaluate (1), the following steps were followed:

1. Cluster centroids: The cluster centroids were initially selected using the random function:

   $$\mu_i = L + \mathcal{U}(0,1) \times (U - L) \tag{2}$$

   where μ_i denotes the centroid value of cluster C_i, with $\mu_i \in \mathbb{R}^d$ and $\mathbb{R}^d$ indicates the d-dimensional real vector space. Here, $\mathcal{U}(0,1)$ denotes a random number generated from a uniform distribution over the interval $[0,1]$. The minimum value of the generator range (i.e. lower boundary) is denoted as L and the maximum value of the range as U (i.e. the upper boundary of the solution space). Therefore, μ_i in (2) can be computed for $i = 1, 2, \ldots, k$ based on a random number generated uniformly within the boundaries L and U. Equation (2) is able to ensure that cluster centroids are uniformly distributed based on the use of the random number drawn from the uniform distribution $\mathcal{U}(0,1)$. This is crucial for ensuring that the centroids are distributed uniformly across the solution space defined by the lower boundary L and the upper boundary U. The function $\mathcal{U}(0,1)$ generates a random number with equal probability within the range 0 and 1. This characteristic of the uniform distribution renders it appropriate for the purpose of evenly distributing cluster centroids. Therefore, when this uniformly distributed random number is scaled by the difference between U and L, and then offset by L, the resulting value μ_i is thus guaranteed to be uniformly distributed within the defined boundaries. This method allows for the initial placement of cluster centroids to cover the entire range of the solution space uniformly, which is essential for the effective partitioning of the data into clusters.
2. Cluster Assignment: Each data point was then assigned to the nearest centroid using Euclidean distance as the measure of assignment as follows:

   $$C_i = \{x \mid \|x - \mu_i\| \le \|x - \mu_j\|, \forall j \neq i\} \tag{3}$$

Thus, after obtaining μ_i and C_i from (2) and (3), respectively, we evaluated W as defined in (1). The objective was to identify the set of cluster centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}$ that minimizes W:

$$\min_{\mu_1, \mu_2, \ldots, \mu_k} W \tag{4}$$

Consequently, it is worth noting that various clustering algorithms typically optimize (1) using distinct strategies for updating centroids. For instance, the k-means algorithm calculates the subsequent centroid for each cluster as the mean of the data points within the assigned cluster as follows:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{5}$$

for $i = 1, 2, \ldots, k$. Conversely, metaheuristics would iteratively update their centroids according to their unique search processes and operators. Nevertheless, the overarching goal is to create clusters that are internally cohesive, thus maximizing the similarity among elements within each cluster.

3.2 Datasets

In this study, we synthetically generated two distinct classes of datasets for evaluation purposes, namely compact-isolated and overlapping-intermingled cluster datasets. We describe these datasets as follows:

3.2.1 Compact-isolated dataset

Compact-isolated clustered datasets, characterized by high intra-cluster similarity, serve as an optimal testing environment for algorithmic evaluation in simpler scenarios. These datasets are particularly useful for benchmarking, offering a robust baseline for algorithmic comparison. Failure of an algorithm to effectively cluster such elementary datasets will likely indicate poor performance in more complex cases. Additionally, the straightforward nature of the results that can be obtained in this case will make interpretation much simpler, thus allowing for a more direct assessment of algorithmic fitness relative to visual clustering outcomes.

For our experimental framework, we examined datasets incorporating cluster distributions of varying complexities comprising 4, 9, and 16 clusters with node sizes of 400, 900, and 1600 data points, respectively. These experimental datasets are visually represented in Fig. 1a–c.

In order to generate the datasets of Fig. 1a–c, the following mathematical procedure was adopted:

1. *Cluster Center Generation* For each cluster, a distinct center (c_{xi}, c_{yi}) was generated in a 2D space. It was ensured that the distance between any two centers is sufficiently large to maintain isolation between clusters. This distance was controlled by a parameter d_{min}, ensuring that for any two centers (c_{xi}, c_{yi}) and (c_{xj}, c_{yj}), the distance $\sqrt{(c_{xi} - c_{xj})^2 + (c_{yi} - c_{yj})^2} \geq d_{min}$.
2. *Data Point Generation* For each cluster, we generated data points around its center using a Gaussian distribution with a small standard deviation (σ). This ensured that the data points were closely packed around the center. For a cluster centered at (c_{xi}, c_{yi}), each data point (x, y) was generated as $x = c_{xi} + \epsilon_x$ and $y = c_{yi} + \epsilon_y$, where $\epsilon_x, \epsilon_y \sim N(0, \sigma^2)$.
3. *Dataset Size and Cluster Proportion* We then allocated the total number of data points (400, 900, 1600) among the clusters (4, 9, 16) evenly based on the predetermined proportion to mimic real-world data distributions.

3.2.2 Overlapping-intermingled dataset

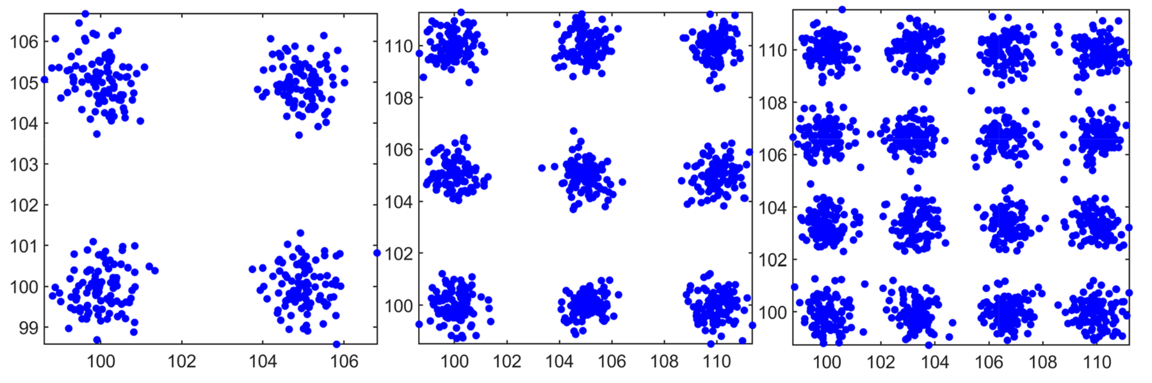
We focused on overlapping-intermingled clusters, which inherently exhibit low cluster purity. This feature serves as a challenging testbed to evaluate the robustness of various algorithms in dealing with noise, outliers, and varying cluster characteristics such as density, shape, and size. Moreover, these complex datasets more closely mirror real-world scenarios, suggesting that algorithms effective in this context may be particularly well-adapted for complex real-world applications.

Similar to the compact cluster case, we analyzed a 2-D solution space populated by randomly generated nodes. The datasets were constructed such that their non-obvious separation of clusters will necessitate the need for robust algorithms for accurate cluster identification. The cluster distributions were varied across 4, 9, and 16 clusters, maintaining a consistent data size per use-case similar to the compact-isolated scenario. Visual representations of these datasets are provided in Fig. 2a–c, respectively. The procedure used to generate these datasets is described as follows:

1. *Cluster Center Generation* Similar to the compact-isolated datasets, we generated distinct centers for each cluster. However, for overlapping datasets, we chose a smaller d_{min} to allow for closer proximity between some of the cluster centers, promoting overlap.
2. *Data Point Generation with Overlap* We then increased the standard deviation (σ) of the Gaussian distribution used for generating data points. This allowed for a broader spread of data points around each center, increasing the likelihood of overlap between clusters. For clusters intended to overlap significantly, we further reduced the distance d_{min} between their centers.

3.3 Candidate algorithms deployed in the HM schemes

Prior to developing the HM schemes, we performed an initial experiment using the simulated datasets detailed in Sect. 3.2 to identify the specific algorithms to be deployed in the HM schemes from among a collection of several candidate metaheuristics. The candidate metaheuristics



(a) 4 clusters - 400 data points (b) 9 clusters - 900 data points (c) 16 clusters - 1600 data points

Fig. 1 Compact-isolated datasets

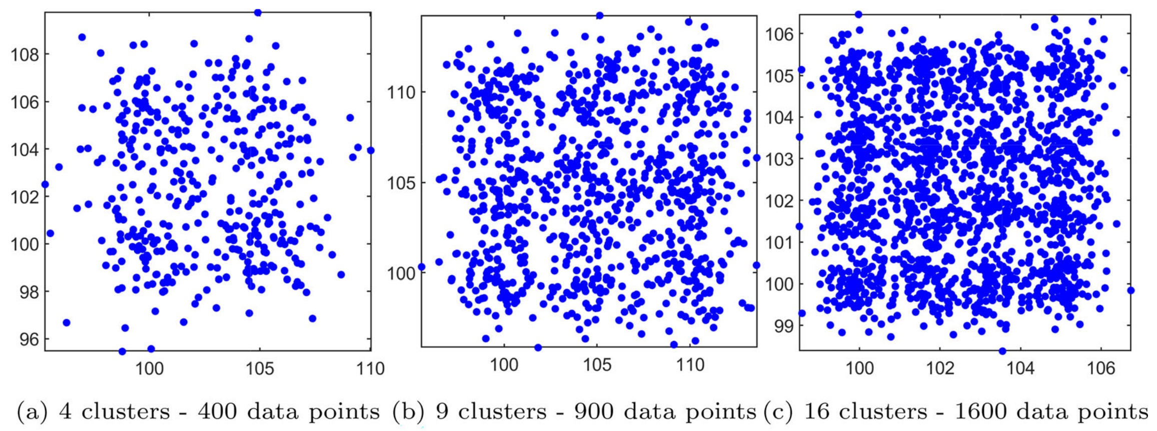


Fig. 2 Overlapping-intermingled datasets

included the covariance matrix adaptation evolution strategy (CMA-ES), cuckoo search optimization (CSO), differential evolution (DE), artificial bee colony (ABC), firefly algorithm (FFA), genetic algorithm (GA), grey wolf optimization (GWO), particle swarm optimization (PSO), whale optimization algorithm (WOA), and the established k-means algorithm. Thereafter, these algorithms were assessed based on their respective fitness values (using (1)) and their evolutionary trends. The results of these experiments are presented in Sect. 5.2 from which an informed selection of the k-means, CMA-ES, and DE algorithms was determined for the HM schemes. In the next subsections, we present a brief description of these chosen algorithms as follows:

3.3.1 k-means

The k-means algorithm is a widely-used clustering method that partitions a dataset into k distinct, non-overlapping subsets (or clusters) [20]. The aim of the algorithm is to minimize the within-cluster sum of distance W (see (1)), effectively finding centroids that are representative of the categories in a multi-dimensional feature space.

In our study, the k-means algorithm was selected for its rapid convergence, which was considered particularly beneficial for use in serial hybrid configurations. The k-means algorithm was used in the following way: given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ where each x_i is a d-dimensional vector, k-means was used to partition the n observations into $k \leq n$ sets $C = \{C_1, C_2, \ldots, C_k\}$ to minimize equation (1). Then, the algorithm was deployed as follows:

1. *Initialization* k initial centroids were chosen, one for each cluster. This was done randomly using (2).
2. *Assignment* Then, each data point x was assigned to the nearest centroid μ_i using (3), thus becoming a member of cluster C_i.
3. *Update* New centroids (μ_i) were then computed as the mean of all points x in C_i using (5).
4. *Convergence* The assignment and update steps were then repeated until the centroids no longer change significantly, at which point the algorithm was considered to have converged, and the final clusters collated.

3.3.2 CMA-ES

The CMA-ES is a stochastic optimization algorithm particularly effective for solving non-linear, high-dimensional optimization problems [21]. One of its distinguishing features is its ability to adapt the search distribution in a way that accounts for the underlying structure of the optimization landscape. From our findings in Sect. 5.2, the CMA-ES and DE algorithms demonstrated superior long-term fitness performance, thus resulting in markedly more accurate clustering outcomes across all use-cases as compared to the other metaheuristics examined in this study.

Operation-wise, the CMA-ES algorithm operates by maintaining a population of candidate solutions μ (i.e. the centroids), which are sampled from a multivariate Gaussian distribution characterized by a mean $\mathbf{m}$ and a covariance matrix $\mathbf{C}$.

$$\mu_i \sim \mathcal{N}(\mathbf{m}, \mathbf{C}), \quad i = 1, 2, \ldots, P \tag{6}$$

Here, P is the population size. After evaluation of the objective function in (1) for each candidate, a selection mechanism is applied to choose the μ best-performing candidates, denoted $\mu_{\text{best},1}, \mu_{\text{best},2}, \ldots, \mu_{\text{best},\mu}$.

The mean $\mathbf{m}$ and the covariance matrix $\mathbf{C}$ are then updated according to the following equations:

1. Update the mean:

$$\mathbf{m}_{\text{new}} = \sum_{i=1}^{\mu} w_i \mu_{\text{best,i}} \tag{7}$$

where w_i are the weights that sum up to 1, typically normalized so $\sum_{i=1}^{\mu} w_i = 1$.

2. Update the covariance matrix:

$$\mathbf{C}_{\text{new}} = (1 - c_1 - c_\mu)\mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^{\mu} w_i (\mu_{\text{best,i}} - \mathbf{m})(\mu_{\text{best,i}} - \mathbf{m})^T \tag{8}$$

where c_1 and c_μ are the learning rate parameters, and $\mathbf{p}_c$ is the evolution path, which accounts for the history of past updates.

These updated **m** and **C** parameters are then used for the next iteration of the algorithm to sample a new set of candidate solutions, thus iteratively improving the optimization.

3.3.3 DE

DE is a stochastic optimization algorithm primarily used for solving continuous optimization problems [22]. It belongs to the family of evolutionary algorithms and operates through a population of candidate solutions. DE employs mutation, crossover, and selection operations to evolve the population towards an optimal or near-optimal solution. It is used in the following algorithmic steps:

1. *Initialization* It generates an initial population $\mathcal{U} = \{\mu_1, \mu_2, \ldots, \mu_P\}$ of centroids using (2), where P is the population size and each μ_i is a d-dimensional real vector in the search space.
2. *Mutation* For each target vector μ_i, a mutant vector v_i is generated using the formula:

$$v_i = \mu_{r1} + F \times (\mu_{r2} - \mu_{r3}) \tag{9}$$

Here, μ_{r1}, μ_{r2}, and μ_{r3} are randomly selected vectors from the current population, distinct from each other and from μ_i, and F is the scaling factor.

3. *Crossover* The mutant vector v_i is then mixed with the target vector μ_i to create a trial vector u_i. This is done using a crossover operation, typically defined as:

$$u_{ij} = \begin{cases} v_{ij} & \text{if } \text{rand}(0,1) \leq CR \text{ or } j = \text{randint}(1, D) \\ \mu_{ij} & \text{otherwise} \end{cases} \tag{10}$$

Here, CR is the crossover rate, D is the number of dimensions, and j is the jth dimension.

4. *Selection* The trial vector u_i then competes against the target vector μ_i based on a fitness function W in (1). The one with the better fitness survives into the next generation:

$$\mu_i^{t+1} = \begin{cases} u_i & \text{if } f(u_i) \leq f(\mu_i) \\ \mu_i & \text{otherwise} \end{cases} \tag{11}$$

5. *Termination* Repeat steps 2–4 until a termination criterion (e.g., maximum number of generations or a satisfactory fitness level) is met.

This iterative process effectively explores and exploits the search space, thus steering the population towards optimal or near-optimal solutions.

3.4 Hybrid schemes

In this section, we describe the algorithms proposed for implementing the HM schemes, building upon the constituent algorithms determined and described in Sect. 3.3. It should be noted that we limited our schemes to bi-algorithmic hybrids, as increasing the number of constituent algorithms beyond two algorithms only yields greater computational and memory overhead without significant improvements in accuracy [23]. We describe the different HM schemes as follows:

3.4.1 Serial hybrid scheme

A high-level block representation of the SHS considered in our study is shown in Fig. 3. The scheme presents an architecture that employs two serially configured algorithms: the first initializes and partially refines cluster centroids, forwarding the intermediate solutions to the second algorithm via a feedforward mechanism. The second algorithm finalizes the refinement until a predefined termination criterion is met, subsequently producing an output.

Considering Fig. 3, the k-means algorithm was selected as Algorithm A due to its rapid convergence capabilities, albeit often to sub-optimal solutions under varying conditions. Algorithm B was implemented using either CMA-ES or DE, resulting in two distinct SHS configurations: k-means combined with CMA-ES (denoted as CAKS) and k-means paired with DE (denoted as DEKS). A detailed representation of the SHS algorithm is presented in Algorithm 1.

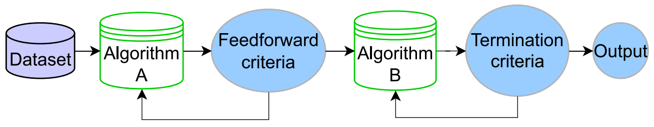
Algorithm 1 Serial hybrid scheme (SHS)

```
Inputs: $X$: Input data points, $K$: Number of clusters, $N$: Number of data points, $U$ and
    $L$: Solution boundaries, $TFE$: total fitness evaluations, $f_c$: Feedforward percentage, $\{\bullet\}$:
    All parameters of the CMA-ES and DE algorithms, $P$: Population size
Outputs: $\mathcal{U}_{best} = \{\mu_1, \mu_2, ..., \mu_K\}$: Final cluster centroids, $C_{best} = \{C_1, C_2, ..., C_K\}$: Final
    cluster datapoints
 1: $FE \leftarrow 0$
 2: Compute the feedforward criteria $FFC$ using (12)
    Initialization:
 3: for k = 1 to K do
 4:   $\mu_k \leftarrow$ Randomly generate a centroid using (2)
 5: end for
    Algorithm A: k-means clustering
 6: while $FE \leq FFC$ do
 7:   for k = 1 to K do
 8:     Assign data points to their nearest centroid using (3)
 9:     Save assigned cluster as $C_k$
10:     Update centroid $\mu_k$ using (5)
11:     Compute fitness of new centroids $C_k$ using (1)
12:     Update fitness evaluation: $FE \leftarrow FE + 1$
13:   end for
14:   Output of k-means is $\mathcal{U}_{kmeans} = \{\mu_1, \mu_2, ..., \mu_K\}$
15: end while
    Algorithm B: metaheuristics Fine-tuning
16: Generate $P - 1$ population of centroids $\mathcal{U}_{P-1,K}$ using (6)
17: Merge $\mathcal{U}_{kmeans}$ and $\mathcal{U}_{P-1,K}$ to form $P$ population
18: while $FE \leq TFE$ do
19:   for i = 1 to P do
20:     if Algorithm 2 $\leftarrow$ CMA-ES then
21:       Update centroids using (7) - (8)
22:       Ensure boundary constraints using (13)
23:       Evaluate fitness of new centroids using (1)
24:       Update fitness evaluation: $FE \leftarrow FE + 1$
25:       Save best centroids $\mathcal{U}_{best}$
26:       Save best clusters $C_{best}$
27:     else if Algorithm 2 $\leftarrow$ DE then
28:       Update centroids using (9) - (11)
29:       Ensure boundary constraints using (13)
30:       Evaluate fitness of new centroids using (1)
31:       Update fitness evaluation: $FE \leftarrow FE + 1$
32:       Save best centroids $\mathcal{U}_{best}$
33:       Save best clusters $C_{best}$
34:     end if
35:   end for
36: end while
37: return Final clusters are outputted as $C_{best}$ and their corresponding centroids as $\mathcal{U}_{best}$
```

The SHS algorithm initializes by specifying the requisite input parameters, including the input data X, target cluster count K, solution space boundaries L and U, total fitness evaluation *TFE*, and feedforward percentage f_c. The algorithm's initialization phase occurs between steps 3 and 4 of Algorithm 1, employing a uniform random number generator to establish K initial centroids.

The subsequent clustering phase leverages k-means (Algorithm A), using the feedforward criteria (*FFC*) as its termination condition. If *FFC* remains unsatisfied (see step 6), the algorithm clusters data points based on their nearest centroids and updates these centroids by averaging the associated data points, as detailed in (5). Following centroid recalculation, their fitness is assessed through the function given in (1). The algorithm then updates the *FE* counter, continuously comparing it to *FFC* until the condition is met.

Fig. 3 Serial hybrid model



The output of the k-means algorithm serves as input to the metaheuristics in the second phase (Algorithm B), for which we employed either DE or CMA-ES. Before initiating Algorithm B, the metaheuristics generates $P-1$ new centroids and combines them with the k-means output to form a population of size P. The metaheuristics then evolves centroids and data point assignments following its inherent search processes, as outlined in Sects. 3.3.2 and 3.3.3 for both DE and CMA-ES, and within steps 20–35 of Algorithm 1. The CBC method is periodically invoked (see steps 22 and 29) to confine centroid values within the feasible search space.

The SHS concludes by outputting the final centroids and their corresponding data clusters, determined by the *TFE* termination criterion. These results are then subsequently made available for further analytical purposes.

SHS: Time complexity The time complexity (TC) of the SHS is described as follows. In the initialization phase denoted by step 3–5 of Algorithm 1, a TC of $\mathcal{O}(K)$ is incurred, where K is the number of the cluster centroids defined in the inputs. The TC of updating the population denoted in steps 6–15 is $\mathcal{O}(K)$, where it involves the iteration of the population until the defined fitness evaluation number is reached. Steps 18–36 perform fine tuning of the parameters, which depends on the number of iterations required until reaching the total fitness evaluations, whose TC is defined as $\mathcal{O}(\mathcal{TFE} * \mathcal{P})$. Therefore, the overall TC of the SHS is given as $\mathcal{O}(2K + TFE * P)$, which asymptotically reduces to $\mathcal{O}(TFE * P)$ since $TFE * P \gg K$. This implies that the SHS scales according the population size and total number of fitness evaluations.

3.4.2 Parallel hybrid scheme

We considered two variants of the parallel hybrid scheme (PHS), namely PHS with coupled and decoupled feedback mechanism as shown in Figs. 4 and 5, respectively. A general description of both variants is given as follows:

Coupled feedback mechanism As illustrated in Fig. 4, the PHS with coupled feedback mechanism (PHS-CFM) employs a common loop that influences the behavior of both constituent algorithms. This means that both algorithms communicate and adapt based on the performance and progress of each algorithm. In this case, both algorithms start with their initial solutions or populations. Then they run concurrently, performing their optimization steps independently. At regular intervals or after a certain number of iterations, both algorithms exchange information about their best solutions i.e. their fitness values via a sort and merge process. Therefore, based on the feedback received from both algorithms, each algorithm then adapts its search process. For example, if one algorithm discovers a promising region of the search space, the other may focus its efforts on exploring that region. Then, both algorithms continue their respective iterations, incorporating feedback from each other, and continuous influencing their respective strategies as the optimization process progresses until the termination criteria is satisfied.

Decoupled feedback mechanism In the PHS with decoupled feedback mechanism (PHS-DCFM), both algorithms operate independently without direct communication or influence from the other algorithm as depicted in Fig. 5. In this case, each algorithm optimizes its solution or population without being aware of the progress of the other algorithm. However, a common termination criteria block ensures that information collected from each algorithm is fed through individual and separate feedback mechanisms to the respective algorithms. This common termination block thus provides a global view of the optimization process, hence enabling the scheme to terminate accordingly.

In examining both PHS variants, we envisaged that the PHS-CFM model will allow the two algorithms to mutually steer themselves towards promising solutions, thereby enhancing the model's speed of convergence and the quality of exploration. However, we anticipated that communication and coordination between both algorithms could introduce weaker solutions, which may not always lead to improved results. Whereas, in the PHS-DCFM model, both algorithms operate independently, which can be advantageous since the behavior of one algorithm should not influence the other, thus allowing for better isolation and control. However, the lack of influence between both algorithms could limit the ability to explore better regions thus impacting the quality of convergence and overall fitness performance of the PHS-DCFM. Consequently, a comprehensive analysis of both variants within the context of clustering was considered pertinent particularly in the absence of such studies in the literature, thus motivating the present study.

Algorithm 2 Parallel hybrid scheme (PHS)

```
Inputs: Same as in Algorithm 1 except $f_c$
Outputs: $\mathcal{U}_{best} = \{\mu_1, ..., \mu_K\}$; $C_{best} = \{C_1, ..., C_K\}$
 1: Algorithm A ← k-means
 2: Algorithm B ← CMA-ES or DE
 3: $FE \leftarrow 0$; $W_{best} \leftarrow \infty$; $\mathcal{U}_{best} \leftarrow [\ ]$; $C_{best} \leftarrow [\ ]$
    Initialization
 4: for k-means: k = 1 to K do
 5:   $\mu_k^{Alg.A} \leftarrow$ Randomly generate a centroid using (2)
 6: end for
 7: for either CMA-ES or DE: i = 1 to P-1 do
 8:   for k = 1 to K do
 9:     $\mu_k^{Alg.B} \leftarrow$ Randomly generate a centroid using (2)
10:     Compute fitness $W_k^{Alg.B}$ of $\mu_k^{Alg.B}$ using (1)
11:     Update fitness evaluation: $FE \leftarrow FE + 1$
12:     Save: $\mathcal{U}_{best} \leftarrow \mu_k^{Alg.B}$ if $W_k^{Alg.B} < W_{best}$
13:   end for
14: end for
15: while $FE \leq TFE$ do
16:   for k-means: k = 1 to K do
17:     Assign data points to nearest centroid using (3)
18:     Save assigned cluster as $C_k$
19:     Update centroid $\mu_k^{Alg.A}$ using (5)
20:     Evaluate fitness $W_k^{Alg.A}$ of $\mu_k^{Alg.A}$ using (1)
21:     Update fitness evaluation: $FE \leftarrow FE + 1$
22:     $\mathcal{U}_{best} \leftarrow \mu_k^{Alg.A}$; $C_{best} \leftarrow C_k$ if $W_k^{Alg.A} < W_{best}$
23:   end for
24:   if PHS-CFM then
25:     Merge: $\mu_k^{Alg.B} \leftarrow \mu_k^{Alg.A} + \mu_k^{Alg.B}$
26:     Sort $\mu_k^{Alg.B}$ in ascending order
27:     Select minimum $\mu_k^{Alg.A} \leftarrow \min\{\mu_k^{Alg.B}\}$
28:   else if PHS-DCFM then
29:     Dont merge: maintain $\mu_k^{Alg.A}$ and $\mu_k^{Alg.B}$
30:   end if
31:   for either CMA-ES or DE: i = 1 to P do
32:     for k = 1 to K do
33:       if Algorithm B ← CMA-ES then
34:         Update centroid $\mu_k^{Alg.B}$ using (7) - (8)
35:         Apply CBC for $\mu_k^{Alg.B}$ using (13)
36:       else if Algorithm B ← DE then
37:         Update centroids $\mu_k^{Alg.B}$ using (9) - (11)
38:         Apply CBC for $\mu_k^{Alg.B}$ using (13)
39:       end if
40:       Assign data points to nearest centroid using (3)
41:       Save assigned cluster as $C_k$
42:       Evaluate fitness $W_k^{Alg.B}$ of $\mu_k^{Alg.B}$ using (1)
43:       Update fitness evaluation: $FE \leftarrow FE + 1$
44:       $\mathcal{U}_{best} \leftarrow \mu_k^{Alg.A}$; $C_{best} \leftarrow C_k$ if $W_k^{Alg.A} < W_{best}$
45:     end for
46:   end for
47: end while
48: return Final clusters $C_{best}$ and centroids $\mathcal{U}_{best}$
```

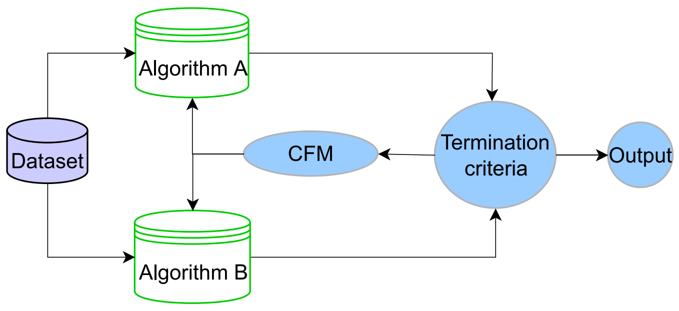


Fig. 4 Parallel hybrid scheme with Coupled feedback mechanism

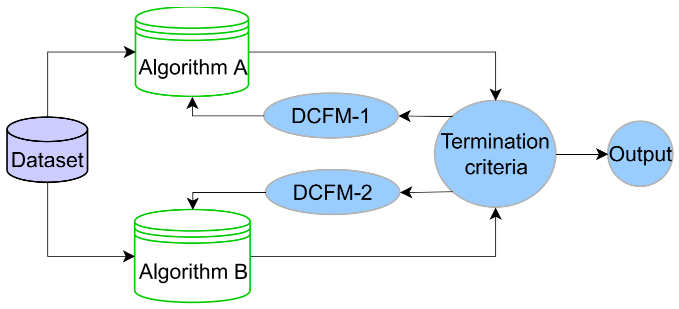


Fig. 5 Parallel hybrid scheme with Decoupled feedback mechanism

We present the internal procedures of the PHS scheme in Algorithm 2. While the input parameters largely mirror those of the SHS, the PHS algorithm uniquely omits the necessity for a feedforward percentage. In the PHS framework, Algorithm A is configured as k-means, while Algorithm B is either CMA-ES or DE. Importantly, the sequence in which these algorithms are arranged is inconsequential, as long as they operate based on their specific search processes.

The PHS begins by initializing the parameters as outlined in step 1 of Algorithm 2. The fitness evaluation *FE* counter is initialized to zero, and the initial best fitness value is set to negative infinity due to the minimization nature of the problem. Thus, a smaller fitness value in (1) indicates superior clustering performance. Memory allocated for the optimal centroid and the related clustered datasets is cleared initially. Then the subsequent steps of Algorithm 2 involve initializing both the k-means algorithm and either CMA-ES or DE by producing random initial centroids between steps 4 and 13. For k-means, a single solution representing the K clusters is generated, with each centroid having d-dimensions. In contrast, for CMA-ES or DE, a population P of initial centroids is produced. For the PHS-CFM model, a population size of $P-1$ is generated, as it will be combined with the k-means solution, resulting in a total population size of P. For the PHS-DCFM model, a full population size P is generated because each algorithm operates with its distinct feedback loop.

The algorithm continually evaluates the solutions generated by both k-means and either CMA-ES or DE, subject to the termination criteria specified in step 15. Upon meeting this criteria, the algorithm ceases execution and returns the results. This ongoing evaluation is facilitated by the *FE* counter at steps 11, 21, and 43 of Algorithm 2. However, should the constraints remain unviolated, the algorithm independently advances with k-means computations in steps 16–23 and CMA-ES or DE in steps 31–45. Furthermore, between steps 24–30, either the CFM or DCFM mechanisms are enforced. Then, execution persists until the termination conditions are met, at which point the optimal centroids and corresponding clustered data points are generated in step 48 and outputted. Based on the described approach for developing the hybrid schemes, Table 1 summarizes the eight hybrid configurations evaluated in this study to enhance clustering performance.

PHS: Time Complexity In contrast to the SHS, the TC of the PHS is slightly different since the configuration of the constituent algorithms and the feedback mechanism employed play an essential role. The TC of PHS with CFM and DCFM is obtained as follows. In the initialization phase (steps 4–14), the TC is $\mathcal{O}((P-1)\times K)$ and $\mathcal{O}(K)$ for the metaheuristics and k-means, respectively. In steps 24–30, the algorithm performs the selection of the feedback mechanism, whose TC can be obtained as follows: steps 25–27 perform the sorting and merging operation using CFM, whose TC after each iteration is increased by $\mathcal{O}(K\log K)$, whereas steps 28–30, where DCFM is employed, the TC remains as without the feedback mechanisms since the achieved solution from both constituent algorithms are maintained separately and no merging operation is performed. The PHS uses steps 16–47 (excluding the selection of feedback mechanism in steps 24–30) to update the population, and the TC differs based on the constituent algorithms employed. For example, the TC when k-means is employed can be formulated as $\mathcal{O}(K)$. On the contrary, when metaheuristics are employed the TC is $\mathcal{O}(P \times K)$. In summary, the overall TC for PHS with CFM is given as $\mathcal{O}(K + (P-1) \times K + TFE \times (K + P \times K) + TFE \times K \log K)$ whereas for PHS with DCFM we obtain $\mathcal{O}(K + (P-1) \times K + TFE \times (K + P \times K))$, where K, P, and TFE denote the number of clusters, population size, and total number of fitness evaluations, respectively. This clearly demonstrates that the PHS is indeed slower than the SHS by the additional order of $\mathcal{O}(K + P \times K)$ in the DCFM option.

3.4.3 Termination criteria and boundary control method used in the HMs

a. *Termination and feedforward criteria* The termination and feedforward criteria used across the hybrid schemes is based on the total fitness evaluations (*TFE*). The *TFE* is a stopping criterion widely used in the literature to control the number of fitness evaluations performed during an optimization process [24]. The *TFE* metric measures the number of fitness function evaluations required by an algorithm to find the optimal or near-optimal solution. This metric provides insights into the effectiveness of the algorithm in exploring the search space and converging to the optimal solution.

   The idea behind using *TFE* as a stopping criterion was to limit the computational budget for optimization and ensure that all the clustering algorithms were terminated fairly and equally after a predefined number of fitness evaluations. In general, it is accepted that this approach helps in controlling the optimization process, especially when the actual fitness landscape or convergence behavior of the problem is unknown [25]. We computed *TFE* algorithmically as follows:

1. *Initialization* We set an initial counter for fitness evaluations *FE* to zero (see step 1 in Algorithm 1).
2. *Optimization loop* In each iteration or generation of the algorithm, we incremented the *FE* counter by the number of fitness evaluations performed in that iteration (see steps 12, 24, and 30 in Algorithm 1).
3. *Termination criteria* We then checked whether *FE* has reached or exceeded the predefined maximum number of fitness evaluations, denoted as *TFE* (see step 18 in Algorithm). If it has, the optimization algorithm then terminates. Otherwise, the process continues.
4. *Feedforward criteria* Furthermore, the feedforward criteria (FFC) in the serial hybrid scheme (SHS) is computed as follows:

$$FFC = (TFE \times f_c)/100 \tag{12}$$

where f_c is the percentage of the total number of *TFE* that will be consumed by Algorithm 1 (i.e. k-means) in the SHS. This parameter can be used to control how long the k-means should be executed before outputting its results to Algorithm 2 (i.e DE or CMA-ES) for further refinement.

We note that the choice of *TFE* depends on various factors, including the available computational resources, the complexity of the optimization problem, and the desired trade-off between exploration and exploitation. Setting *TFE* too low may result in premature convergence, while setting it too high may lead to excessive computational time. Therefore, tuning this parameter is an important aspect of using *TFE* as a stopping criterion in metaheuristicss.

b. *Clipping boundary control method* We used the clipping boundary control (CBC) method to ensure that candidate solutions generated during the optimization process in the different HM schemes remain within the feasible solution space defined by the problem's boundaries. This is to ensure that the clustering centroids do not drift outside the solution space. CBC works by checking whether a generated

Table 1 Proposed hybrid configurations

Hybrid scheme	Feedback mechanism	Configuration		Hybrid acronym
SHS	Feedforward	k-means	CMA-ES	CAKS
		k-means	DE	DEKS
PHS	CFM	k-means	CMA-ES	CAKP-CFM
		k-means	DE	DEKP-CFM
		CMA-ES	DE	CADEP-CFM
	DCFM	k-means	CMA-ES	CAKP-DCFM
		k-means	DE	DEKP-DCFM
		CMA-ES	DE	CADEP-DCFM

candidate solution (i.e. centroid) violates any of the problem's boundary constraints (i.e. L and U). If a constraint is violated, the method "clips" or adjusts the solution to bring it back within the valid range while preserving as much of the solution's quality as possible.

In our hybrid schemes, the CBC was implemented for a single variable as follows [26]:

$$x_{i,j} = \begin{cases} U_j, & \text{if } (x_{i,j} > U_j) \\ L_j, & \text{if } (x_{i,j} < L_j) \\ x_{i,j} & \text{otherwise.} \end{cases} \quad (13)$$

where, $x_{i,j}$ represents the current i *th* solution (i.e. centroid) of the j *th* dimension, and the pair U and L represent the upper and lower bounds of the j *th* dimension, respectively. By using this method, the hybrid schemes ensure that the generated solutions remain within the problem's feasible solution space, even if the algorithm explores regions outside the bounds. It helps prevent infeasible solutions and maintains the integrity of the optimization process. Next, we describe the SHS algorithm.

4 Application to base station placement in smart grids

4.1 Problem definition

To demonstrate the use of our proposed hybrid clustering schemes, we examined the optimal placement of multiple edge base stations (BSs) within a large-scale smart grid network, as depicted in Fig. 6. The network spans a substantial geographic area (potentially a city or province) and incorporates home controllers (e.g. smart meters) installed in residential buildings. These home controllers (i.e. nodes) adopt wireless communication technologies to transmit data to an edge base station, which subsequently relays the data to a central edge data center for processing and action, contingent upon the state of the smart grid.

The objective is to determine the optimal locations for BSs to maximize network coverage while minimizing associated costs. We presume that the coordinates of a given set of home controllers, denoted by $\mathcal{P}$ are known *a priori* and indexed by $i = 1, 2, \ldots, |\mathcal{P}|$.

In contrast, the locations of $\mathcal{N}$ BSs remain undetermined. Thus, let $p_i = (x_i, y_i)$ represent the coordinates of different i home controllers, which are assumed to be randomly distributed within a given spatial domain. Although the initial locations of BSs are uncertain, it is assumed that an optimal position exists within the two-dimensional region $\mathcal{D} \in \mathbb{R}^2$, where the placement of a BS j can maximize the coverage of controllers. Here, $\mathcal{D}$ demarcates the spatial boundaries of the solution space.

Hence, the location of a BSs is denoted as

$$n_j = (x_j, y_j) \; \forall j = 1, 2, \ldots, n \quad (14)$$

where j represents the index of a BS and $(x_j, y_j) \in \mathcal{D}$ the geographical coordinates of the BS. To mitigate the computational burden of channel estimation, we assume that each home controller i can associate with at most one BS in the domain $\mathcal{D}$. Thus, the coverage of a home controller at location p_i by a BS at (x_j, y_j) is determined using the association indicator described in [27] as follows:

$$z_{ij} = \begin{cases} 1, & \text{if } d(p_i, n_j) \le r_c \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

where the coverage radius of an associated BS is designated as r_c, and $d(p_i, n_j)$ represents the Euclidean distance between them, which can be computed as:

$$d(p_i, n_j) := ||p_i, n_j|| \quad (16)$$

where $\| \cdot \|$ denotes the Euclidean distance between home controller p_i and edge BS n_j. We note that for controller i to satisfy coverage criteria as specified in (15), either of two conditions must be fulfilled:

1. The received signal-to-noise ratio (SNR) from the associated base station (BS) must be greater than or equal to the SNR from any other BSs within set $\mathcal{D}$, or
2. The distance $d(p_i, n_j)$ must be minimized relative to the distances to other BSs.

For the first condition, the equation for free-space power reception at the BS can be used as [28]:

$$P_r = \frac{P_t G_t G_r \lambda^2}{L (4\pi)^2 d^2} \quad (17)$$

where P_t denotes the transmission power, d represents the distance between the BS and home controller, λ denotes the wavelength of the transmitted signal, L stands for the path

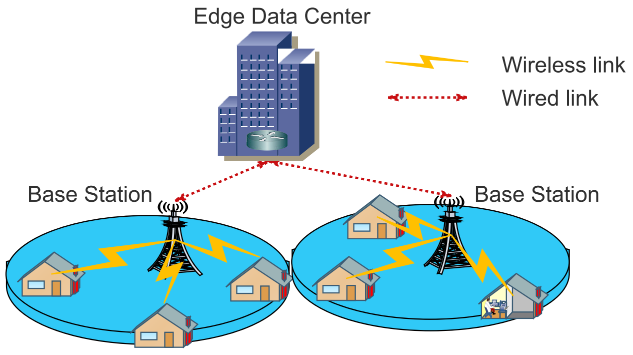


Fig. 6 Illustration of an edge-enabled smart grid network

loss, G_t characterizes the antenna gain of the BS, and G_r refers to the antenna gain of the home controller. However, in our study, we used the second condition, which is based on the network coverage indicator described in the next section.

4.2 Network coverage indicator

For the objective of BS placement, we considered the coverage performance using the network coverage indicator (NCI) represented by the number of connected nodes (*CN*) and non-connected nodes (*NCN*). The *CN* represents the count of nodes within the coverage area of a BS, while *NCN* denotes nodes outside this coverage.

NCI calculation We calculated the NCI measures (i.e. both *CN* and *NCN*) algorithmically by first setting the coverage radius d of each BS (often determined by the maximum transmit power of the BS), and subsequently counting the nodes that lie within this radius. This was achieved as follows: let d_j be the coverage radius of a BS j, and let D_{ij} represent the distance from the BS j to a home controller i. The home controller i is considered to be within the coverage area of the BS j and counted into the set *CN* if the following condition is met:

$$D_{ij} \leq d_j \tag{18}$$

Otherwise, if D_{ij} is greater than d_j, the home controller i is outside the coverage area of the BS j and counted into the set *NCN*. Consequently, by using the NCI measure, our objective is to maximize the network coverage hence minimizing communication costs, further ensuring that all nodes (i.e. home controllers) can reliably communicate their data to the network infrastructure via the BS. This problem can be addressed using the HM schemes by formulating it as an optimization problem with specific objective functions and constraints. Below is the mathematical formulation used for solving BS placement problem using the HM clustering approach:

4.2.1 Placement objective function

The primary objective is to maximize the coverage of each smart grid home controller, i in the smart grid network. In this case, the coverage can be defined as the proportion of nodes within the communication range of at least one base station, which can be mathematically formulated as

$$\max \mathcal{C} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

where N is the total number of nodes, and x_i is a binary variable indicating whether node i is within the communication range of at least one BS (1 if covered, 0 otherwise). The value of x_i is obtained algorithmically using the NCI approach described above.

4.2.2 Placement constraints

In terms of the BS placement constraints, the following were considered:

1. Each node i must be within the communication range of at least one BS j to be considered covered as given in (18), where d_j remains the maximum communication range of a BS.
2. The number of nodes connected to each BS must not exceed its capacity:

   $$\sum_{i=1}^{N} D_{ij} \leq \beta_j, \quad \forall j$$

   where β_j is the maximum number of nodes (i.e. capacity) that can be connected to BS j.
3. The total number of BSs deployed may be limited by budget or logistical considerations:

   $$\sum_{j=1}^{M} b_j \leq \mathcal{B}$$

   where b_j is a binary variable indicating whether BS j is deployed (1 if deployed, 0 otherwise), and $\mathcal{B}$ is the maximum number of BSs allowed, which corresponds to the number of cluster centroids inputted into the clustering method. However, in our study, we set $M = \mathcal{B}$.

5 Results and discussion

In this section, we report on the performance of the different hybrid schemes for both the clustering and optimal BS placement problems. First, we enumerate the parameters employed in the different algorithms under consideration. Subsequently, we present our findings from the algorithm selection phase, in which we examined ten standalone algorithms (k-means, ABC, CMA-ES, CSO, DE, FFA, GA, GWO, PSO, and WOA) as detailed in Sect. 3.3. Evaluation was conducted using the datasets described in Sect. 3.2. Finally, we report on the comparative results obtained for the various hybrid schemes as well as in the context of the optimal BS placement problem.

5.1 Parameter settings

We used the grid-search method to determine the parameter settings for the individual algorithms considered in our study. The results from the well-known grid search process are reported in Table 2. All methods, including the hybrid and individual algorithms were implemented in MATLAB R2022b on a PC with an Intel Core i7 dual processor and 32 GB RAM. The clipping boundary control was used in all algorithms to ensure that solutions were maintained within the search space constraints.

From Table 2, the common inputs were determined and justified as follows: a population size of 5 was chosen to expedite the computational rates across the various algorithms by drawing a small sample from the population. This small population size was then compensated for by using a large *TFE* value of 10,000, which ensured that sufficient examination time was given for algorithms to converge to their global solutions without bias. To ensure a reasonable number of experimental repetitions without overwhelming computational resources, the number of Monte Carlo trials was set at 100.

Regarding the parameters of the individual metaheuristics, the grid search method was used as follows: we defined a range of values for each parameter, thus creating a discrete set of options for testing purposes via the grid search method. This facilitated evaluating every possible combination within the grid. Subsequently, a series of experiments were conducted, each testing the metaheuristic with a different parameter combination from the grid. An analysis of experimental results was conducted for which the parameter combination yielding optimal performance were identifed based on *TFE*, fitness values, and timing. The results of the grid search are thus reported in Table 2, and these values were maintained for the same constituent algorithms deployed in the different HM schemes.

5.2 Selection of the constituent algorithms

This section presents the outcomes of the selection analysis described in Sect. 3.3, which aimed to identify suitable algorithms for inclusion in our hybrid schemes. The standalone algorithms were evaluated using the compact-isolated and overlapping-intermingled datasets described in Sect. 3.2.

The results for the compact-isolated datasets characterized by 4, 9, and 16 clusters are depicted in Fig. 7a–c, respectively. The mean fitness values were calculated over 100 Monte Carlo trials for each algorithm, with the total fitness evaluations (*TFE*) fixed at 10,000. These analyses served dual objectives: firstly, to assess the convergence speed of each algorithm with respect to the *TFE*, and secondly, to gauge the accuracy of the final output at $TFE = 10,000$. This dual assessment allows for the identification of the algorithm that not only converges most rapidly but also achieves the highest overall fitness at the conclusion of the trials.

In Fig. 7, the k-means, CMA-ES, and DE algorithms demonstrated better performance relative to the other methods for the multiple use-cases comprising 4, 9, and 16 clusters. This better performance is quantified by the lower mean fitness values, indicative of minimal intra-cluster distances, achieved by these algorithms in the context of a minimization clustering problem. Specifically, the k-means algorithm exhibited rapid convergence, as evidenced by the horizontal trajectories in the graphs, although it settled at sub-optimal solutions. This property renders the k-means algorithm a viable candidate for hybridization with more proficient algorithms like the CMA-ES and DE algorithms.

Furthermore, both the CMA-ES and DE algorithms performed better than the k-means in terms of solution quality across all use-cases for the compact-isolated datasets. Moreover, Fig. 7a–c reveals a consistent increase in the mean fitness values (depicted on the y-axis) as the number of clusters and data size was progressively increased. This trend corroborates the increasing complexity of the problem space, indicating that it becomes more difficult for all the algorithms as the cluster number and sizes were increased from Fig. 7a–c. This validates the suitability of our experiments as a benchmark for algorithmic evaluation as considered in our study.

As evidenced in Fig. 8, a comparative analysis of the various algorithms as applied to the overlapping-intermingled datasets revealed a consistently better performance from the k-means, CMA-ES, and DE algorithms. Although the reasons for such improved performances by these algorithms remains challenging owing to the heterogeneity of their respective search mechanisms, nevertheless, our study does elucidate patterns of efficacy under varying cluster numbers and sizes. Furthermore, we observed that the k-means algorithm exhibited rapid convergence and robust performance consistently across the different datasets examined here. Consequently, we can affirm that across the diverse datasets, the trio of the k-means, CMA-ES, and DE algorithms all consistently delivered high-quality clustering results (i.e., low mean fitness values), hence substantiating their inclusion in our proposed hybrid schemes.

5.3 Performance of the hybrid schemes

The hybrid configurations outlined in Table 1 were assessed using the benchmark datasets from Sect. 3.2. The outcomes for each use case can be found in Fig. 9 for compacted-isolated datasets and Fig. 10 for the overlapping-intermingled datasets.

Upon the analysis of the mean fitness values over 100 Monte Carlo trials, we found no statistically significant differences in performance for the low cluster number dataset (i.e. 4 clusters, refer to Fig. 9a). However,

Table 2 Parameters settings across the different algorithms

Common inputs	Population size = 5, *TFE* = 10000, Monte Carlo trials = 100, $f_c = 5$, Number of clusters = Data dependent
ABC	Abandonment limit = 24, Acceleration coefficient upper bound = 1, Onlooker Bees = population size
CMA-ES	Except for the population size, all operational parameters in CMA-ES are internally configured, rendering the method essentially user-parameter-free.
CSO	Discovery of alien eggs = 0.2
DE	Crossover probability = 0.2, Lower bound of scaling factor = 0.2, Upper bound of scaling factor = 0.8
FFA	Light absorption coefficient = 1, Attraction coefficient base value = 2, Mutation coefficient = 0.2, Mutation coefficient damping ratio = 0.98
GA	Crossover percentage = 0.8, Number of offsprings = 4, Selection rate = 0.8, Selection operator = Roulette wheel, Mutation percentage = 0.2, Crossover type = Single point crossover, Elitism rate = 0.5
GWO	Beyond the standard inputs, the GWO technique is devoid of additional user-defined parameters
PSO	Inertia weight (adaptive) = (0.1,1.1), Self adjustment weight = 1.5, Acceleration coefficients = Uniformly distributed (0,1) random vector, Social adjustment weight = 1.5
WOA	Beyond the standard inputs, the GWO technique is devoid of additional user-defined parameters

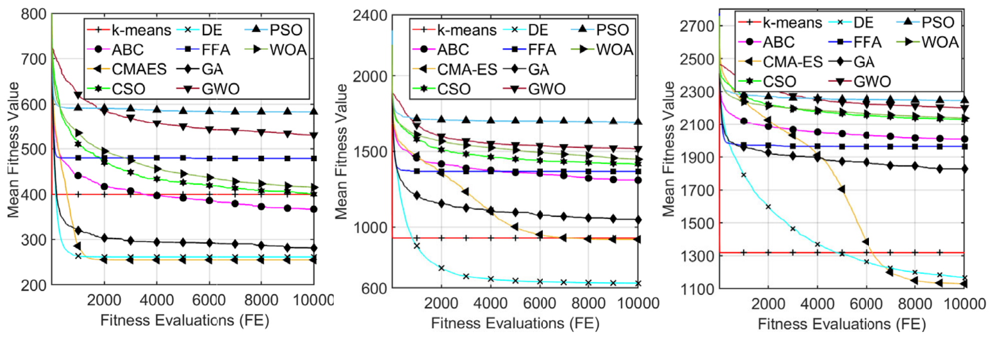


Fig. 7 Compact-isolated use-case showing the fitness evolution of the different standalone candidate algorithms

discernible variations in mean fitness performance across the various hybrid algorithms began to manifest as the cluster number was increased, as illustrated in Fig. 9b, c. However, among the evaluated algorithms, the CADEP-DCFM, DEKP-DCFM, and DEKS schemes consistently outperformed other methods across the different datasets. It is noteworthy that while the CADEP-DCFM scheme converged to optimal mean fitness values eventually, its rate of convergence was conspicuously slower in the 16-cluster dataset.

In the compact-isolated scenario of Fig. 9, the hybrid schemes with decoupled feedback mechanism outperformed those with the coupled feedback mechanism. On the other hand, the serial configurations employing the DE and k-means algorithms exhibited comparable performance to their parallel counterparts. Notably, we observed that the serial configurations consistently demonstrated a lower variance in fitness values as compared to their parallel counterparts.

In the analysis of the overlapping-intermingled datasets, as illustrated in Fig. 10, the CADEP-DCFM scheme demonstrated robust performance across the varied datasets. Notably, in scenarios with a high number of clusters, as shown in Fig. 10c, CADEP-DCFM exhibited satisfactory results in the long run, albeit at a slower fitness evaluation rate. In comparison, the DEKS scheme showcased competitive performance against its parallel counterparts while surpassing its CAKS analog in most cases.

Furthermore, our analysis reveals that the hybrid parallel configurations, particularly those incorporating two metaheuristicss with the decoupled feedback mechanism, exhibited the best performance in terms of clustering accuracy when evaluated over the extended range of the *TFE*. However, this configuration lacks in computational speed (i.e. achieving good clustering results within small fitness evaluation rates) as compared to the other methods. Consequently, we argue that for applications prioritizing execution speed over clustering accuracy, the serial hybrid configuration employing the k-means and DE algorithm

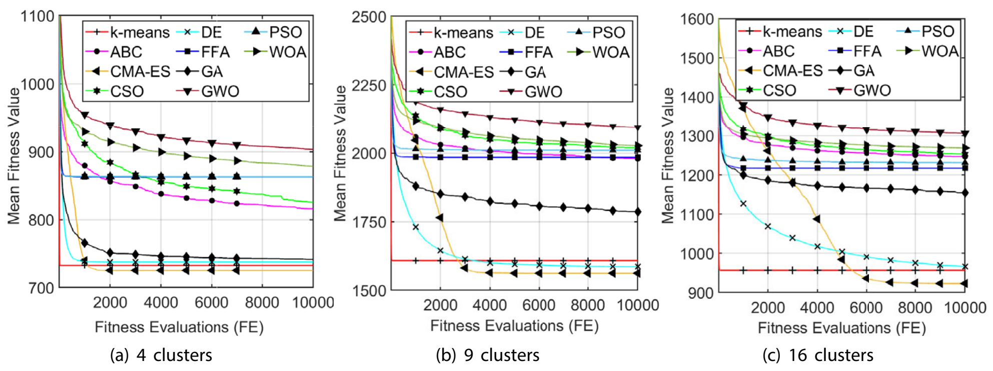


Fig. 8 Overlapping-intermingled use-case showing the fitness evolution of the different standalone candidate algorithms

should be recommended. Conversely, if accuracy is of paramount importance, then the CADEP-DCFM algorithm will be the most suitable choice. It is also noteworthy that the application of the CMA-ES algorithm generally resulted in slower convergence rates, thereby affecting the overall efficiency of such hybrid configurations that depend on it.

Finally, upon comparing the performances of the standalone methods presented in Figs. 7 and 8 to the hybrid schemes in Figs. 9 and 10, it is evident that the hybrid schemes outperformed the standalone methods. This can be confirmed following the summary statistics presented in Table 3, which shows the mean and standard deviation of the fitness values across all the clustering techniques considered in our study. This results underscore the advantage of harnessing the individual strengths of the original methods when constructing hybrid schemes. Further discussions in the subsequent section will focus on the timing performances of these methods.

5.4 Physical timing performance

The computational efficiency of the different algorithms was quantified based on their average execution times, as reported in Table 4. These results were collated following simulations conducted on a computer with specifications mentioned in Sect. 5.1. The values reported were also calculated based on 100 Monte Carlo simulations per algorithm, and conducted across all datasets presented in Sect. 3.2. These timing measurements were equally benchmarked by terminating all algorithms at $TFE = 10,000$ and the results obtained are as shown in Table 4.

Our findings demonstrated a positive correlation between the timing performance of each algorithm and the data sizes across the types of datasets examined. This confirms that computational time typically increases with an increase in the data size, thereby justifying the variety of datasets considered in the benchmark experiments of this study.

Furthermore, from Table 4, we observed no significant timing difference among the methods tested across the different datasets and cluster configurations. Specifically, the time difference between the fastest original method (i.e. WOA) and the slowest hybrid method (i.e. CAKP-DCFM) is approximately 2 s for the 4-cluster scenario. In the 9 and 16 cluster scenarios, the maximum deviations recorded was 3 and 14 s, respectively. Thus, by using the *TFE* as the stopping criterion, we ensured a fair comparison among all algorithms, thus avoiding any undue advantage. This further corroborates the argument in [24], which supports the efficacy of using *TFE* to assess both timing and accuracy performance. In addition, the results of Table 4 further

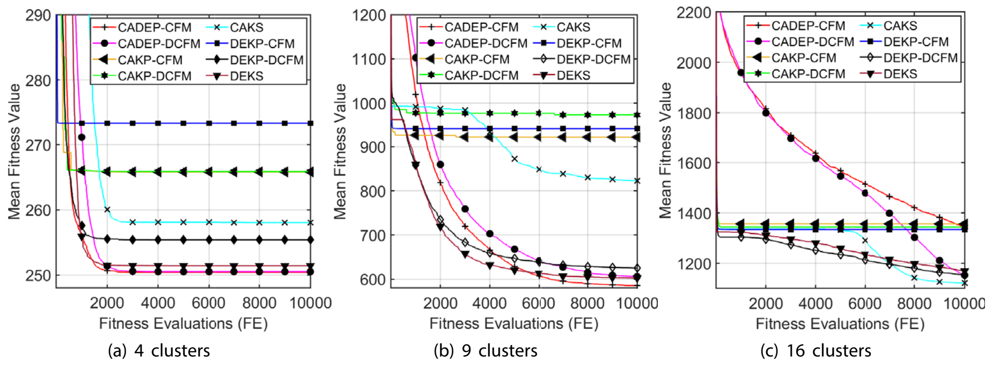


Fig. 9 Fitness evolution of hybrid schemes for the compact-isolated datasets

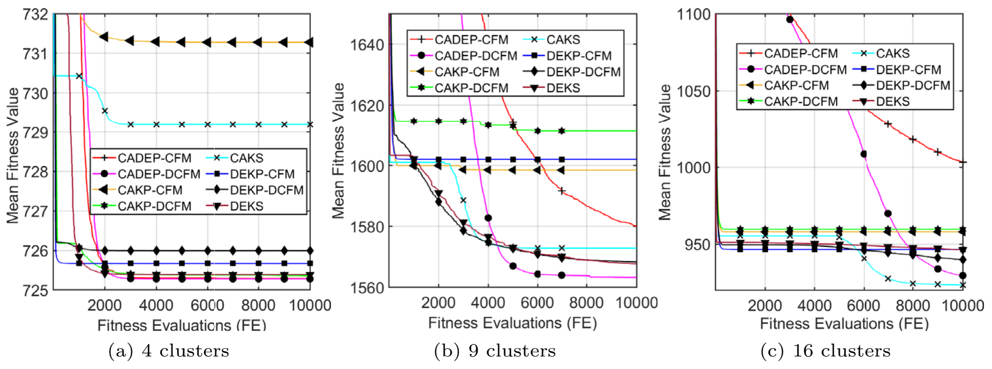


Fig. 10 Fitness evolution of hybrid schemes for the overlapping-intermingled datasets

suggest that hybrid approaches can effectively rival original methods in terms of execution time as long as they are controlled using the *TFE* measure.

5.5 Comparison to state-of-the-art HM schemes

In this section, we discuss the proposed HM schemes in comparison to existing methods towards highlighting the advantages of our research and schemes. Firstly, in Table 5 we present the diverse configurations and feedback mechanisms utilized by various HM schemes documented in the literature. Upon reviewing Table 5, it becomes evident that a majority of existing schemes do not incorporate the DCFM approach. This observation underscores the absence of DCFM-based analysis in existing schemes and highlights its integration within our proposed schemes.

Furthermore, many existing HM methods typically initiate clustering with k-means, and then refine the solutions using other metaheuristics such as the PSO algorithm. However, as we have demonstrated in Sect. 5.2, both the PSO, GA, and other approaches yielded unsatisfactory results across the datasets examined in this study. This highlights the importance of considering diverse data characteristics, revealing that algorithms like PSO and GA often struggle in clustering tasks, potentially due to their limited ability to explore the problem space effectively. Consequently, there is a need for enhancing their operational procedures. Additionally, in many of the HM-based clustering articles in the literature, most authors often failed to provide the results of their pre-selection experiments and analyses prior to the development of their HM schemes. In contrast, such detailed pre-selection experiments were undertaken in our study with sample methods drawn from across both swarm and evolutionary-based algorithms. Hence, this enabled us to select the best performing metaheuristics for our HM schemes based on the characteristics of our diverse datasets.

Consequently, we have investigated the efficacy of the DE and CMA-ES algorithms based on their noted efficiency and robustness in addressing various clustering scenarios, as evidenced by the datasets employed in our research. Within most parallel schemes in Table 5, the CFM method emerges as a popular choice, likely owing to its straightforward implementation as compared to the DCFM. However, our results analysis presented in Table 3 reveals that the PHS-DCFM scheme exhibited superior performance, particularly in scenarios involving a large number of clusters. Being a scheme seldom explored by prior methodologies, as indicated in Table 5, we have demonstrated a critical advantage of our scheme as well as other configurations adopted in this research.

5.6 Edge base station placement

The various hybrid algorithms were also employed to optimally position four base stations (BSs) with the aim of maximizing network coverage in the clustered settlements depicted in Fig. 11. The resulting BS locations exhibited negligible variations, which was attributed to the aggregation of the top-performing outcomes from 100 Monte Carlo simulations. Consequently, the observed consistency in results was expected, as our emphasis was on placement accuracy rather than convergence speed.

Furthermore, we assumed that the datasets outlined in Sect. 3.2 depicted networks with clustered nodes necessitating optimal BS placement. Subsequently, we employed all methods considered in our study to determine the optimal locations of multiple BSs. The outcomes were analyzed in terms of the connectivity metrics, namely CN for connected nodes and NCN for non-connected nodes, and these are reported in Table 6, with the best methods emphasized in bold.

From Table 6, it is evident that the hybrid schemes consistently outperformed the original methods across all

Table 3 Mean and standard deviation of fitness values across all methods and datasets

Algorithms	Compact-isolated			Overlapping-intermingled		
	K = 4	K = 9	K = 16	K = 4	K = 9	K = 16
ABC	401.02 ± 33.0	1378.0 ± 72.94	2056.99 ± 58.18	841.95 ± 28.21	2014.06 ± 44.55	1266.40 ± 24.91
CADE-CFM	257.63 ± 3.75	726.92 ± 80.68	1623.70 ± 143.63	732.83 ± 3.56	1673.47 ± 46.68	1081.0 ± 36.98
CADE-DCFM	263.26 ± 6.79	766.21 ± 93.27	1569.66 ± 140.02	733.91 ± 3.83	1646.09 ± 34.97	1051.97 ± 33.67
CAKP-CFM	267.41 ± 77.56	923.67 ± 247.57	1357.77 ± 182.77	731.90 ± 34.15	1600.43 ± 72.76	958.98 ± 34.42
CAKP-DCFM	267.51 ± 78.03	977.53 ± 220.52	1345.26 ± 202.95	726.05 ± 0.85	1614.33 ± 77.77	960.58 ± 31.67
CAKS	271.77 ± 65.74	902.39 ± 250.63	1266.37 ± 155.64	729.54 ± 27.84	1582.0 ± 58.06	943.50 ± 23.60
CMA-ES	273.48 ±36.55	1115.20 ± 252.73	1666.42 ± 119.40	739.62 ± 5.01	1672.40 ± 31.19	1081.20 ± 28.40
CSO	445.29 ± 53.36	1476.90 ± 90.05	2187.50 ± 70.60	862.72 ± 38.57	2063.57 ± 55.64	1280.07 ± 31.60
DE	267.68 ± 28.40	714.64 ± 80.22	1405.29 ± 159.02	741.90 ± 24.52	1634.05 ± 39.51	1029.46 ± 36.89
DEKP-CFM	273.87 ± 90.41	942.15 ± 241.38	1335.30 ± 166.13	726.30 ± 0.68	1603.69 ± 65.52	947.72 ± 30.0
DEKP-DCFM	258.34 ± 20.17	693.17 ± 85.44	1235 ± 143.30	726.66 ± 0.78	1580.04 ± 29.59	947.59 ± 23.93
DEKS	258.74 ± 14.48	671.15 ± 94.82	1255.61 ± 145.58	726.03 ± 2.75	1579.13 ± 35.28	949.66 ± 28.14
FFA	480.83 ± 110.39	1371.55 ± 161.51	1970.96 ± 131.02	863.87 ± 71.08	1987.05 ± 111.91	1219.20 ± 57.53
GA	301.48 ± 88.52	1120.45 ± 152.37	1890.76 ± 126.69	752.47 ± 15.35	1830.48 ± 65.30	1176.40 ± 43.99
GWO	565.52 ± 140.88	1573.14 ± 182.89	2264.84 ± 133.82	926.40 ± 76.65	2138.84 ± 128.63	1333.33 ± 61.95
k-means	399.36 ± 188.1	929.12 ± 253.67	1319.44 ± 145.75	732.62 ± 36.08	1608.69 ± 71.80	956.78 ± 32.17
PSO	586.99 ± 103.67	1706.27 ± 152.98	2261.38 ± 133.15	863.84 ± 57.26	2012.79 ± 90.34	1235.31 ± 44.92
WOA	462.72 ± 160.56	1522.71 ± 179.0	2186.85 ± 141.25	901.57 ± 79.96	2067.95 ± 124.94	1284.81 ± 59.56

Table 4 Average execution time (in seconds)

Algorithms	Compact-isolated			Overlapping-intermingled		
	K = 4	K = 9	K = 16	K = 4	K = 9	K = 16
ABC	16.2	50.5	127.09	15.53	50.99	117.21
CADEP-CFM	16.68	50.9	135.15	16.13	51.76	123.46
CADEP-DCFM	16.74	50.28	136.74	15.97	50.98	123.62
CAKP-CFM	17.31	52.35	141.09	16.59	52.76	127.66
CAKP-DCFM	17.41	52.84	139.47	16.54	52.54	125.69
CAKS	17.27	53.5	139.55	16.58	53.37	126.27
CMA-ES	17.36	51.95	135.58	16.64	52.61	121.58
CSO	16.13	49.83	133.2	15.42	50.47	116.08
DE	16.15	49.83	131.93	15.55	50.58	115.49
DEKP-CFM	16.43	49.91	138.83	15.77	50.74	121.63
DEKP-DCFM	16.26	49.64	138	15.59	50.63	120.83
DEKS	16.33	49.75	135.48	15.65	50.33	120.71
FFA	16.09	50.12	132.78	15.56	51.03	116.1
GA	16.54	49.96	130.25	15.99	51	118.01
GWO	15.78	49.45	127.24	15.21	50.13	117.09
K-Means	15.82	49.49	129.76	15.01	49.18	116.78
PSO	15.88	49.7	130.07	15.58	50.84	117.89
WOA	15.46	49.89	126.17	14.92	50.36	117.04

Table 5 Configuration and feedback mechanisms of different HM schemes in comparison to the proposed HMs

References	Configuration		Constituent algorithm			Feedback mechanism	
	Serial	Parallel	K-Means	DE	CMA-ES	CFM	DCFM
MBCO [8]	✔	✗	✔	✗	✗	✗	✗
Hybrid FFA [9]	✔	✗	✔	✗	✗	✗	✗
ATPSO [12]	✔	✗	✔	✗	✗	✗	✗
SA-PSO-GK++ [13]	✔	✗	✔	✗	✗	✗	✗
Hybrid PSO [14]	✗	✔	✗	✗	✗	✔	✗
MPGO [15]	✗	✔	✗	✗	✗	✔	✗
PSOGSA [17]	✗	✔	✗	✗	✗	✔	✗
ED-DE [18]	✗	✔	✗	✔	✗	✔	✗
Hybrid BFO [19]	✗	✔	✗	✗	✗	✔	✗
SHS (Proposed)	✔	✗	✔	✔	✔	✗	✗
PHS-CFM (Proposed)	✗	✔	✔	✔	✔	✔	✗
PHS-DCFM (Proposed)	✗	✔	✔	✔	✔	✗	✔

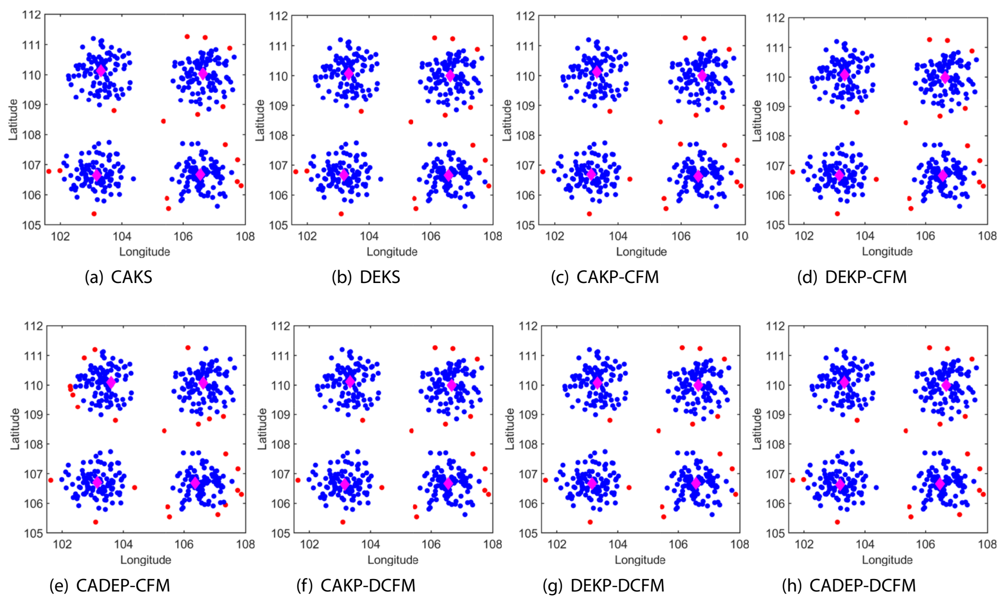


Fig. 11 Base station placement using the hybrid schemes (the purple diamonds represent the location of the BS; the red nodes are the non-connected nodes, whereas the blue nodes are the connected nodes.)

the datasets. However, within the hybrid schemes, although the parallel methods exhibited a slightly better performance over the serial schemes, nevertheless the difference was marginal. In the case of the larger cluster networks, it was expected that more nodes will be unconnected since only one BS was assigned per node cluster. We note that for more realistic network configurations, additional BSs may be required to improve coverage, or an increase in BS transmitter power may be necessitated. However, the primary focus of our study was to elucidate the viability of the proposed hybrid clustering schemes for cost-effective network planning, especially in scenarios where the cost of deploying BSs must be minimized.

6 Conclusion

In this paper, we have proposed two algorithms to study eight different HM-based clustering schemes, which integrate pre-selected metaheuristics and the k-means algorithm to improve clustering performance. Our approach involved the use of a well-defined objective function, investigation across diverse datasets, and a well-documented selection process to determine the candidate algorithms for the hybrid schemes. We further examined the performance of both the coupled and decoupled feedback mechanisms across the different hybrid configurations. Our findings indicate that the parallel configurations with a decoupled feedback mechanism demonstrated better

Table 6 Base station coverage performance (*CN* connected nodes and *N-CN* non-connected nodes)

Class	Algorithms	Compact-isolated						Overlapping-intermingled					
		K = 4		K = 9		K = 16		K = 4		K = 9		K = 16	
		CN	N-CN	CN	N-CN	CN	N-CN	CN	N-CN	CN	N-CN	CN	N-CN
Serial hybrid	CAKS	376	24	835	65	1524	76	306	94	704	196	1509	91
	DEKS	376	24	833	67	1527	73	306	94	705	195	1587	13
Parallel hybrid	CADE-CFM	376	24	835	65	1493	107	306	94	704	196	1508	92
	CADE-DCFM	376	24	835	65	1518	82	306	94	706	194	1536	64
	CAKP-CFM	376	24	833	67	1527	73	306	94	705	195	1550	50
	CAKP-DCFM	376	24	837	63	1523	77	306	94	704	196	1590	10
	DEKP-CFM	376	24	837	63	1528	72	306	94	701	199	1504	96
	DEKP-DCFM	378	22	836	64	1529	71	304	96	704	196	1503	97
Original methods	ABC	200	200	126	774	671	929	120	280	350	550	1116	484
	CMA-ES	356	44	785	115	1483	117	306	94	654	246	1498	102
	CSO	342	58	497	403	867	733	303	97	566	334	1296	304
	DE	346	54	796	104	1463	137	306	94	651	249	1449	151
	FFA	366	34	647	253	1313	287	286	114	684	216	1215	300
	GA	375	25	745	155	1219	381	299	101	677	223	1292	308
	GWO	363	37	596	304	945	655	291	109	630	270	1280	320
	k-means	363	37	826	74	1498	102	306	94	643	257	1446	154
	PSO	285	115	372	528	982	618	298	102	614	286	1266	334
	WOA	372	28	517	383	1014	586	291	109	618	282	1204	396

clustering accuracy over an extended range of total fitness evaluations (*TFE*), albeit at a slower convergence rate as compared to the serial-based methods. Therefore, for scenarios prioritizing speed, we can recommend the serial hybrid configuration, for example, using k-means and the DE algorithm. Conversely, for applications where accuracy is crucial, a parallel configuration with the decoupled feedback mechanism may be most appropriate. Importantly, by using the *TFE* as the stopping criterion in our algorithms, we confirm that there were no significant differences in the computational times between the hybrid and individual metaheuristics. Consequently, future works will aim to explore self-adaptive and mutual-adaptive learning mechanisms as well as assess the effects of different initialization and boundary control methods on the clustering performance of these hybrid schemes.

Author contributions D.N. Molokomme conceptualized the research contribution, wrote the methodology, implemented the algorithms using the software, and wrote the original draft of the manuscript. Whereas, A. J. Onumanyi and A. M. Abu-Mahfouz supervised, did formal analysis, and also reviewed and edited the manuscript.

Funding Open access funding provided by University of Pretoria. This research was funded by the Council for Scientific and Industrial Research (CSIR).

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests .

References

1. Azaza, M., Wallin, F.: Smart meter data clustering using consumption indicators: responsibility factor and consumption variability. Energy Procedia **142**, 2236–2242 (2017). https://doi.org/10.1016/j.egypro.2017.12.624

2. Ghosal, A., Nandy, A., Das, A.K., Goswami, S., Panday, M.: A Short Review on Different Clustering Techniques and Their Applications, pp. 69–83. Springer, New York (2019). https://doi.org/10.1007/978-981-13-7403-6_9
3. Yadav, S.A., Poongodi, T.: A novel chain-based clustering for green communication in wireless sensor network. Int. J. Commun. Syst. **36**(13), 5523 (2023). https://doi.org/10.1002/dac.5523
4. Baalamurugan, K.M., Bhanu, S.V.: An efficient clustering scheme for cloud computing problems using metaheuristic algorithms. Clust. Comput. **22**, 12917–12927 (2019). https://doi.org/10.1007/s10586-018-1800-4
5. Mirsadeghi, E., Khodayifar, S.: Hybridizing particle swarm optimization with simulated annealing and differential evolution. Clust. Comput. **24**, 1135–1163 (2021). https://doi.org/10.1007/s10586-020-03179-y
6. Mageshkumar, C., Karthik, S., Arunachalam, V.: Hybrid metaheuristic algorithm for improving the efficiency of data clustering. Clust. Comput. **22**, 435–442 (2019). https://doi.org/10.1007/s10586-018-2242-8
7. Naghavipour, H., Idris, M.Y.I.B., Soon, T.K., Salleh, R.B., Gani, A.: Hybrid metaheuristics using rough sets for qos-aware service composition. IEEE Access **10**, 112609–112628 (2022). https://doi.org/10.1109/access.2022.3213705
8. Das, P., Das, D.K., Dey, S.: A modified bee colony optimization (MBCO) and its hybridization with k-means for an application to data clustering. Appl. Soft Comput. **70**, 590–603 (2018). https://doi.org/10.1016/j.asoc.2018.05.045
9. Xie, H., Zhang, L., Lim, C.P., Yu, Y., Liu, C., Liu, H., Walters, J.: Improving k-means clustering with enhanced firefly algorithms. Appl. Soft Comput. **84**, 105763 (2019). https://doi.org/10.1016/j.asoc.2019.105763
10. Golalipour, K., Akbari, E., Hamidi, S.S., Lee, M., Enayatifar, R.: From clustering to clustering ensemble selection: a review. Eng. Appl. Artif. Intell. **104**, 104388 (2021). https://doi.org/10.1016/j.engappai.2021.104388
11. Zhou, P., Sun, B., Liu, X., Du, L., Li, X.: Active clustering ensemble with self-paced learning. IEEE Trans. Neural Netw. Learn. Syst. (2023). https://doi.org/10.1109/tnnls.2023.3252586
12. Xu, X., Li, J., Zhou, M., Xu, J., Cao, J.: Accelerated two-stage particle swarm optimization for clustering not-well-separated data. IEEE Trans. Syst. Man Cybern. **50**(11), 4212–4223 (2018). https://doi.org/10.1109/tsmc.2018.2839618
13. Abdo, A., Abdelkader, O., Abdel-Hamid, L.: SA-PSO-GK++: a new hybrid clustering approach for analyzing medical data. IEEE Access **12**, 12501–12516 (2024). https://doi.org/10.1109/ACCESS.2024.3350442
14. Cheng, R., Sun, C., Jin, Y.: A multi-swarm evolutionary framework based on a feedback mechanism. In: 2013 IEEE Congress on Evolutionary Computation, pp. 718–724. IEEE (2013). https://doi.org/10.1109/cec.2013.6557639
15. Huang, K.-W., Wu, Z.-X., Peng, H.-W., Tsai, M.-C., Hung, Y.-C., Lu, Y.-C.: Memetic particle gravitation optimization algorithm for solving clustering problems. IEEE Access **7**, 80950–80968 (2019). https://doi.org/10.1109/access.2019.2923979
16. Qtaish, A., Braik, M., Albashish, D., Alshammari, M.T., Alreshidi, A., Alreshidi, E.J.: Optimization of k-means clustering method using hybrid capuchin search algorithm. J. Supercomput. (2023). https://doi.org/10.1007/s11227-023-05540-5
17. Chaudhari, S., Thakare, A., Anter, A.M.: Psogsa: a parallel implementation model for data clustering using new hybrid swarm intelligence and improved machine learning technique. Sustain. Comput. **41**, 100953 (2024). https://doi.org/10.1016/j.suscom.2023.100953
18. Wang, Y., Li, B., Weise, T.: Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. Inf. Sci. **180**(12), 2405–2420 (2010). https://doi.org/10.1016/j.ins.2010.02.015
19. Madhusudhanan, B., Sumathi, P., Karpagam, N.S., Mahesh, A., Suhi, P.A.P.: An hybrid metaheuristic approach for efficient feature selection. Clust. Comput. **22**(Suppl 6), 14541–14549 (2019). https://doi.org/10.1007/s10586-018-2337-2
20. Dias, L.A., Ferreira, J.C., Fernandes, M.A.C.: Parallel implementation of k-means algorithm on FPGA. IEEE Access **8**, 41071–41084 (2020). https://doi.org/10.1109/access.2020.2976900
21. Beyer, H.-G., Sendhoff, B.: Simplify your covariance matrix adaptation evolution strategy. IEEE Trans. Evol. Comput. **21**(5), 746–759 (2017). https://doi.org/10.1109/tevc.2017.2680320
22. Shilaja, C.: Implementation of differential evolution algorithm and its variants for optimal scheduling of distributed generations. Int. J. Commun. Syst. **34**(6), e4318 (2020). https://doi.org/10.1002/dac.4318
23. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998). https://doi.org/10.1109/34.667881
24. Črepinšek, M., Liu, S.-H., Mernik, M.: Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them. Appl. Soft Comput. **19**, 161–170 (2014). https://doi.org/10.1016/j.asoc.2014.02.009
25. Mehrmolaei, S., Keyvanpour, M.R., Savargiv, M.: Metaheuristics on time series clustering problem: theoretical and empirical evaluation. Evol. Intell. **15**(1), 329–348 (2020). https://doi.org/10.1007/s12065-020-00511-8
26. Kadavy, T., Viktorin, A., Kazikova, A., Pluhacek, M., Senkerik, R.: Impact of boundary control methods on bound-constrained optimization benchmarking. IEEE Trans. Evol. Comput. **26**(6), 1271–1280 (2022). https://doi.org/10.1145/3583133.3595849
27. Cao, B., Fan, S., Zhao, J., Tian, S., Zheng, Z., Yan, Y., Yang, P.: Large-scale many-objective deployment optimization of edge servers. IEEE Trans. Intell. Transp. Syst. **22**(6), 3841–3849 (2021). https://doi.org/10.1109/tits.2021.3059455
28. Rappaport, T.S.: Wireless communications-principles and practice, (the book end). Microw. J. **45**(12), 128–129 (2002)

Daisy Nkele Molokomme received her B.Tech and M.Tech in Electrical Engineering from the Department of Electrical and Electronic Engineering Technology, University of Johannesburg, Johannesburg, South Africa, in 2017 and 2021, respectively. She is currently with the Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa while pursuing a Ph.D degree in Electronic Engineering with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South

Africa. Her current research interests include smart transactive microgrids, edge computing, metaheuristic algorithms, and artificial intelligence.

Adeiza James Onumanyi earned his B.Eng. in Electrical and Electronics Engineering from Abubakar Tafawa Balewa University in Bauchi, Nigeria, in 2005, and his M.Eng. and PhD in Communication Engineering from the Federal University of Technology (F.U.T) in Minna, Nigeria, in 2010 and 2014, respectively. Adeiza is a researcher at South Africa's Council for Scientific and Industrial Research (CSIR) in Pretoria. He has several research articles published in peer-reviewed journals and at IEEE flagship conferences. Between 2010 and 2021, he lectured and conducted research at the Department of Telecommunication Engineering, F.U.T, Minna, Nigeria, where he was involved in securing several grants, serving on several organizing committees for various conferences, including IEEE conferences, reviewing several articles for high impact journals, and participating in various technical workshops. Among his research interests are spectrum sensing in cognitive radio, wireless sensor networks, smart transactive microgrids, DC nanogrids, radar systems, image processing, cyber physical systems, and low powered wireless area networks.

Adnan M. Abu-Mahfouz (M'12-SM'17) received his MEng and PhD degrees in computer engineering from the University of Pretoria. He is currently a Chief Researcher and the Centre Manager of the Emerging Digital Technologies for 4IR (EDT4IR) research centre at the Council for Scientific and Industrial Research (CSIR), Extraordinary Professor at University of Pretoria, Professor Extraordinaire at Tshwane University of Technology and Visiting Professor at University of Johannesburg. His research interests are wireless sensor and actuator network, low power wide area networks, software defined wireless sensor network, cognitive radio, network security, network management, sensor/actuator node development. He is a Section Editor-in-Chief at the Journal of Sensor and Actuator Networks, an associate editor at IEEE Access, IEEE Internet of Things and IEEE Transaction on Industrial Informatics, Senior Member of the IEEE and Member of many IEEE Technical Communities.