Clustering Problems for More Useful Benchmarking of Optimization Algorithms

Marcus Gallagher*

School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, 4072. Australia.
marcusg@itee.uq.edu.au

Abstract. This paper analyses the data clustering problem from the continuous black-box optimization point of view and proposes methodological guidelines for a standard benchmark of clustering problem instances. Clustering problems have been used many times in the literature to evaluate evolutionary, metaheuristic and other global optimization algorithms. However much of this work has occurred independently and the various experimental methodologies used have produced results which tend to be incomparable and provide little collective wisdom as to the difficulty of the problems used, or an objective measure for comparing and evaluating the performance of algorithms. This paper surveys some of the clustering literature and results to identify issues relevant for benchmarking. A set of 27 problem instances ranging from 4-D to 40-D and based on three well-known datasets is identified. To establish some pilot results on this benchmark set, experiments are presented for the Covariance Matrix Adaptation-Evolution Strategy and several other standard algorithms. A web-repository has also been created for this problem set to facilitate better experimental evaluations of algorithms.

Keywords: Algorithm Benchmarking, Continuous Black-box Optimization, Clustering

1 Introduction

In evolutionary computation and metaheuristic optimization, an enormous number of algorithms have been developed. Since no algorithm is superior in the theoretical, No Free Lunch sense, in practice the performance differences we observe depend on how well the mechanisms of the algorithm match the structure of the problem landscape. A key step towards understanding the matching between problems and algorithms is to develop better benchmark problems and more rigorous approaches to the experimental analysis of algorithms. Unfortunately, the dominating paradigm in the literature has been to continually develop new algorithm variants and to evaluate these techniques in isolation. For continuous

* The author acknowledges the contribution of the Dagstuhl Theory of Evolutionary Algorithms Seminar 13271 (http://www.dagstuhl.de/13271/) to the work in this paper.

black-box optimization, artificial test functions (e.g. Sphere, Rastrigin, Rosenbrock) have been used hundereds of times, but a question such as"what is the best performance of a black-box optimization algorithm on function f, given 10^5 functions evaluations?" seems to be difficult (if not impossible) to answer using the literature. The situation is even more problematic, because subtle differences in experimental settings in different papers (e.g. using a different bound on the feasible search space) mean that results are often not strictly comparable. Recently, research has begun to focus more on such experimental issues. For example, the Black-Box Optimization Benchmarking (BBOB) problem set[1] resolves many of these issues by standardizing many aspects of the experimental setting. However, it is very important to also evaluate algorithms on real-world problems, since it is difficult to know how well artificial test problems represent real-world problems and hence to what extent algorithm performance on artificial problems is indicative of real-world performance. It can be difficult to use real-world problems for algorithm benchmarking because real problems may require expert domain knowledge to configure, or may come with additional complexities that are not part of the basic optimization algorithm (e.g. complex constraints). Ideally, problems that are real-world "representative" while being convenient for benchmarking should provide a valuable contribution to experimental research practice.

This paper examines data clustering as a useful source of continuous, black-box benchmark problems. In Section 2, the sum of squares clustering problem is defined and its key properties discussed. Clustering problems have previously been used in the literature to test optimization algorithms: Section 4 reviews some of this literature and discusses why it is difficult to compare with previously reported results. A specification is proposed to describe clustering problem instances and a set of problem instances defined (and made available via the web). To establish some baseline results for future comparison, a number of commonly-used algorithms are applied to the clustering problem sets. The experiments are described in Section 5 and Results presented in Section 6. Where possible, the results are also compared with previous results from the literature, revealing some surprising insights. The work is summarised in Section 7.

2 Clustering

The sum of squares clustering problem (see, e.g.[13]) can be stated as follows. Given a set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq I\!R^d$ of n data points, find a set of k cluster centers $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \in I\!R^d$ to minimize:

$$f(\mathcal{C}|\mathcal{X}) = \sum_{i=1}^{n} \sum_{j=1}^{k} b_{i,j} ||\mathbf{x}_i - \mathbf{c}_j||^2$$

where

$$b_{i,j} = \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{c}_j|| = \min_j ||\mathbf{x}_i - \mathbf{c}_j|| \\ 0 & \text{otherwise.} \end{cases}$$

[1] http://coco.gforge.inria.fr/doku.php

The problem variables are the coordinates of the cluster centres in the data space. Let the d-dimensional coordinates of $\mathbf{c}_i = (y_{d(i-1)+1}, y_{d(i-1)+2}, \ldots, y_{di})$, then we have an unconstrained, continuous optimization problem of dimensionality dk:

$$\min f(\mathbf{y}), \mathbf{y} \in \mathbb{R}^{dk}$$

A clustering problem instance is therefore defined by a dataset, $\mathcal{X}$ and a value of k.

An equivalent problem from operations research is the (continuous, uncapacitated) location-allocation problem , also known as facility location or multisource Weber problem [2, 9]. Given a set of *customers* to be serviced by a set of *facilities*, the problem is to position the facilities to optimize a criterion measuring overall service. Under the following conditions:

- the set of customer locations is given by $\mathcal{X}$,
- the set of facility locations be $\mathcal{C}$,
- assuming equal customer weightings, unlimited capacity of facilities to provide service and Euclidean distances between customers and facilities,

the problem then reduces to the sum of squares clustering problem.

Clustering is a fundamental task in machine learning, data analysis and operations research. Finding a global optimum is known to be NP-hard, even in the restricted cases where $d = 2$ or $k = 2$. A large number of algorithms have been proposed for clustering, though there is little doubt that the k-means algorithm is the most widely known and used [11]. From an optimization perspective, k-means is a local iterative optimization algorithm which follows a non-increasing trajectory over f. It is not a black-box algorithm, nevertheless its popularity makes it frequently used in experimental comparative studies. Note also that solving the optimization problem (i.e. locating cluster centres) is often not the final goal of clustering. Further analysis might include studying which data points are assigned to which cluster centre, or producing a classifier, where each cluster represents a class in the data set and the class label of future data points can then be predicted (e.g. using the minimum distance from the cluster centres).

3 Why Use Clustering Problems for Black-Box Optimization Benchmarking?

Clustering problems have a number of properties which suggest that they might provide an extremely useful source of benchmark problems for the evaluation and comparison of algorithms:

- They seem to be generally challenging to solve.
- They are scalable in dimensionality (via d and k).
- They are “real-world” problems in data analysis (i.e. datasets can come from real-world problems).
- They are unconstrained, meaning that black-box algorithms can be readily applied without the need for a constraint-handling mechanism.

– They can be implemented relatively simply and do not require a large amount of problem-domain-specific knowledge to understand.
– The objective function is not expensive to evaluate.

There are currently few (if any) benchmark problem sets that have all of the above properties. This suggests an exciting opportunity to improve on and increase the utility of experimental black-box algorithm evaluation and comparison by building a standardised set of clustering problem instances.

4 Black-box Optimization Approaches to Clustering

Given the fundamental nature of the clustering problem and data analysis, it is not surprising that hundreds of clustering algorithms have been proposed in the literature. At the same time, general-purpose metaheuristics and other optimization techniques have also been applied to clustering problems. This paper does not attempt an exhaustive review of all this work, but rather aims to extract the important issues to be considered in developing a specification of clustering problems for black-box optimization benchmarking.

4.1 Difficulties in Comparing with Previous Results

One of the major difficulties in trying to compare an algorithm with previous work stems from the lack of standard in the way authors present their results. Clustering results are presented in a variety of ways in the literature [13]. While the sum of squares objective function is frequently used, the actual function values (and the number of evaluations made by the algorithm) obtained are sometimes not reported. Instead, measures of cluster shape around the cluster centres produced have been used (e.g. the Rand index is used by Chang et al. to evaluate their genetic algorithm variant [3]). When the intended application is classification, measures such as classification accuracy on the data are used (e.g. Liu et al.[7] evaluate a fuzzy C-means, genetic algorithm based fuzzy C-means and an immunodominance clonal selection fuzzy C-means algorithm in this way).

While clustering problems have been widely used to compare algorithms, the datasets that have been used also vary from paper to paper. Some authors generate artificial datasets with known structure/distributions. There are a large number of benchmark datasets available in machine learning, and different authors select different datasets to use. Focussing on specific datasets would clearly improve the comparability of results for black-box optimization algorithms.

Finally, there are many experimental factors that are not specific to clustering problems that can impact on future comparisons of results. The fundamental performance results are in terms of the best objective function values found (or statistics of such values over multiple trials) and the number of function evaluations used. Presenting results in figures has several advantages, but on the other hand it is often difficult to read off numerical values from a graph for comparison. Full details of the experimental configuration (e.g. algorithm parameter settings, termination criteria, number of repeated trials) are essential to permit fair comparison and reproduction of results.

4.2 Results Selected for Comparison

The literature was further reviewed for experimental results that could be compared in a black-box optimization context. A representative number of approaches were identified:

- Maulik [8] develops a real-valued genetic algorithm (GA) for clustering and compares with k-means.
- Ye and Chen [14] apply particle swarm optimization (PSO), ant colony optimization (ACO) and a honey bees algorithm.
- Kao and Cheng [6] develop an ant colony optimization algorithm and compare it with a previous ACO clustering algorithm (due to Shelokar et al.[10]) and k-means.
- Fathian et al.[5] present a honey bee mating algorithm and compare it with ACO, a GA, Tabu search (TS) and simulated annealing (SA).
- Taherdangkoo et al.[12] propose a blind, naked mole-rats algorithm and compare it with k-means, two GA variants, PSO, ACO, simulated annealing and artificial bee colony algorithms.

These papers have each used different datasets to evaluate and compare algorithms. One problem instance is common across all the papers - these results are compared in Section 6.1.

4.3 Clustering Problem Instances

In the literature, many different datasets have been utilized to evaluate and compare clustering algorithms. Sometimes, authors generate artificial test data with known clustering structure. This can be useful, for example to visualize results. However if the exact dataset used is not available, then results can only be compared qualitatively. Benchmark datasets have also been widely used, such as those from the UCI Machine Learning Repository [1]. In particular, Du Merle et al.[4] used an interior point algorithm to compute approximate global optimum values for problem based on the Iris, Ruspini and German Towns (Spath) datasets. This is useful because we can assess the performance of algorithms relative to the optimal value on these problems. These datasets have also been used in other papers, therefore the following set of problem instances is used:

- The Iris dataset, with $d = 4$, $k = 2, \dots, 10$ and initial search space $[0.1, 7.9]^{dk}$.
- The Ruspini dataset, with $d = 2$, $k = 2, \dots, 10$ and initial search space $[4, 156]^{dk}$.
- The German Towns dataset, with $d = 3$, $k = 2, \dots, 10$ and initial search space $[24.49, 1306024]^{dk}$.

Clustering Problem Specification To be useful for black-box optimization evaluation, a clustering problem should be specified with the following elements:

1. A dataset, $\mathcal{X}$ of dimensionality d.
2. A value of k.
3. An initial bounded search space, which contains the global optimum. This can be done by using the minimum and maximum value in the dataset as the upper and lower bounds of the search space. For simplicity the overall minimum and maximum are used for every variable. A tighter search space could consider the minimum and maximum of each variable independently, however the implementation would be more complex.

To facilitate future use of these problems, a web repository has been created at http://realopt.uqcloud.net/crwr.html. The repository records the specifications of each problem instance, the global optimum (solution vector and objective function value) and a copy of the dataset. This will be extended to record results on these problems from the literature.

5 Experimental Details

To make a comparison and establish some results for the selected clustering problems, the following algorithms were evaluated:

– CMA-ES: the Matlab implementation of the Covariance Matrix Adaptation Evolution Strategy available from https://www.lri.fr/∼hansen/cmaes_inmatlab.html was used with default parameter settings (as recommended, the initial search variance was set to 2/3 of the search space.
– CMA-ES (50,100): the same implementation of CMA-ES but with a (larger) population size of 50.
– NM: the Nelder-Mead simplex algorithm, as implemented in the Matlab `fminsearch` function. Default parameter settings were used, with termination criteria extended so that the algorith ran until a tolerance of change in variables or function values was less than 10^{-10}, or if 3×10^5 function evaluations were reached. The algorithm is initialized at a random point in the search space for each trial.
– RS: uniform random search over the search space. The algorithm was given 10^5 function evaluations.
– KM: the k-means clustering algorithm. Cluster centers were initialized to be randomly selected data points (this is probably the most common method in the literature but there are many other possibilities [11]).

The algorithms were chosen firstly because they can be applied with little setting of internal parameters. In addition: CMA-ES is a well-regarded black-box algorithm; NM is the standard Matlab solver; KM is a standard non-black-box clustering algorithm; RS provides a useful baseline. Each algorithm was run for

50 restarts. Note that the different algorithms ran for different numbers of function evaluations. The intention was not to impose a fixed budget of function evaluations across the algorithms but to allow them to use the amount of resources they request to"converge". Results on these problems can be of interest for any reasonable budget of function evaluations. Future research may choose to focus on a "limited budget" scenario or on finding high quality solutions using a possibly large number of function evaluations. Different algorithm specifications will be more suitable to different budgets of function evaluations and any result that improves upon previous results makes a worthwhile contribution.

6 Results

The experimental results are shown in Tables 1 and 2. Overall, CMA(50,100) gave the best performance, with average values that were closer to the optimal value that the other techniques. It required between 10000 and 50000 function evaluations. CMA used a smaller population size (determined automatically) and used between 2000 and 25000 function evaluations. The results were very similar on some problems (e.g. for Ruspini $n = 4, 6, 10$) but an order of magnitude worse on others (e.g. German, $n = 8-20$). The NM results are considerably worse across the problem sets than CMA(50,100) and worse than CMA for the Iris and Ruspini problems, but (interestingly) better on the German Towns problems. As a local search algorithm, it does however use a much smaller budget of function evaluations: between 1000 and 8000. As a completely non-local algorithm, RS outperforms the standard Matlab solver (NM) on most of the Ruspini problem instances! Finally, KM as a non-black-box solver has a considerable advantage over the other algorithms. It converges very quickly, taking less than 20 iterations/function evaluations across the problems tested. Its performance is relatively good, however CMA(50,100) still provides better performance on all problem instances! This is an impressive result for a black-box solver and experimentally demonstrates that metaheuristics are able to outperform problem-specific algorithms, and that global/population-based search would seem to lead to results that are difficult to obtain with a local/trajectory-based algorithm. With such small requirements for function evaluations, large amount of restarts could be performed for KM. Nevertheless the results here over 50 runs at least indicate that the fitness landscapes of clustering problems contain structure that causes problems for the standard solver in this problem domain.

On most problems (shown with bold), CMA(50,100) found the global optimum on at least one of the 50 trials. The exceptions were some of the larger problems on the German Towns problems. It is an open question to establish results on these problems to see how many functions evaluations are required to locate the global optimum. KM finds the global optimum for around half of the problems (lower dimensions) and CMA and NM do so for some of the smaller problem instances. Figs 1-3 compare the average fitness performance of the algorithms over the problems from each dataset. Results are given as a performance ratio with the global optimum value for each problem instance.

Table 1. Results for clustering problems. Problem instances are defined by a dataset (D), i.e. Iris (I), Ruspini (R) or German towns (G), together with the problem dimensionality (n), where $n = dk$. Shown are mean and standard deviations over 50 trials of each algorithm. The mean and average number of function evaluations ($\#f$) is also shown for each algorithm. A result is in bold if one or more of the 50 trials located the global optimum (f^*) to at least 15 significant figures.

D	n	f^*	CMA(50,100)	CMA(50,100) $\#f$	CMA	CMA $\#f$
I	8	1.52348e02	**1.523480e02(0.0e00)**	1.1168e04(2.9e02)	**2.68733e02(2.2e02)**	2.41780e03(2.1e02)
	12	7.88514e01	**7.885144e01(0.0e00)**	1.7136e04(5.7e02)	**1.96530e02(1.8e02)**	3.77214e03(5.3e02)
	16	5.72285e01	**5.922328e01(5.0e00)**	2.1938e04(7.4e02)	1.26181e02(8.8e01)	4.98944e03(5.9e02)
	20	4.64462e01	**4.904783e01(2.3e00)**	2.5092e04(9.5e02)	1.20999e02(8.9e01)	6.58952e03(9.8e02)
	24	3.90400e01	**4.076697e01(3.5e00)**	2.7466e04(1.3e03)	1.06623e02(8.9e01)	8.39298e03(1.3e03)
	28	3.42982e01	**3.554887e01(1.4e00)**	3.0220e04(1.1e03)	8.90713e01(3.3e01)	9.90306e03(1.2e03)
	32	2.99889e01	**3.232285e01(2.1e00)**	3.3250e04(9.3e02)	8.84456e01(3.3e01)	1.20515e04(1.8e03)
	36	2.77861e01	**2.946070e01(1.9e00)**	3.6432e04(1.4e03)	8.38895e01(2.9e01)	1.46102e04(2.4e03)
	40	2.58341e01	**2.705470e01(1.3e00)**	3.9214e04(1.5e03)	8.00406e01(3.2e01)	1.67537e04(3.0e03)
R	4	8.93378e04	**8.93378e04(0.0e00)**	1.3842e04(4.2e03)	**8.93378e04(0.0e00)**	1.4188e03(3.0e02)
	6	5.10635e04	**5.11278e04(4.4e01)**	1.9382e04(2.9e03)	**5.11094e04(4.8e01)**	3.2726e03(3.2e02)
	8	1.28811e04	**1.28811e04(0.0e00)**	1.6822e04(5.1e03)	**1.66519e04(1.2e04)**	2.8380e03(7.9e02)
	10	1.01267e04	**1.10334e04(6.3e02)**	2.0862e04(4.9e03)	**1.14295e04(1.2e03)**	3.1720e03(2.2e02)
	12	8.57541e03	**8.84859e03(6.8e02)**	2.2022e04(1.1e03)	9.86935e03(9.6e02)	3.8949e03(2.1e02)
	14	7.12620e03	**7.55527e03(7.4e02)**	2.5912e04(2.3e03)	8.76290e03(1.4e03)	4.9135e03(7.2e02)
	16	6.14964e03	**6.43566e03(3.5e02)**	2.8792e04(2.7e03)	7.75958e03(1.5e03)	5.7632e03(6.7e02)
	18	5.18165e03	**5.64378e03(2.0e02)**	3.1892e04(4.3e03)	6.67363e03(1.1e03)	6.1772e03(7.1e02)
	20	4.446.28e03	**4.80930e03(2.5e02)**	3.1822e04(2.4e03)	6.87915e03(1.4e03)	7.2632e03(1.5e03)
G	6	6.02546e11	**6.02547e11(0.0e00)**	1.6282e04(4.7e03)	1.55257e12(8.2e11)	3.6002e03(4.5e02)
	9	2.94506e11	**3.08336e11(2.9e10)**	2.2952e04(5.5e03)	1.04493e12(8.0e11)	5.4150e03(8.9e02)
	12	1.04474e11	**1.42481e11(8.0e10)**	2.8652e04(8.1e03)	1.05834e12(7.8e11)	7.3940e03(1.7e03)
	15	5.97615e10	**7.46629e10(1.5e10)**	3.2272e04(5.0e03)	8.33722e11(7.2e11)	1.0106e04(2.4e03)
	18	3.59085e10	**4.80932e10(9.0e09)**	3.3402e04(1.5e03)	5.99661e11(6.0e11)	1.19768e04(1.4e03)
	21	2.19832e10	4.40172e10(9.4e09)	3.9092e04(2.6e03)	4.31552e11(1.5e11)	1.44593e04(1.8e03)
	24	1.33854e10	**3.11688e10(1.2e10)**	4.0962e04(2.6e03)	4.17723e11(1.6e11)	1.82163e04(1.4e03)
	27	7.80442e09	2.26611e10(1.2e10)	4.6382e04(3.5e03)	4.05634e11(1.8e11)	2.09944e04(1.9e03)
	30	6.44647e09	2.80614e10(1.2e10)	4.9872e04(5.8e03)	4.19464e11(1.7e11)	2.3522e04(3.2e03)

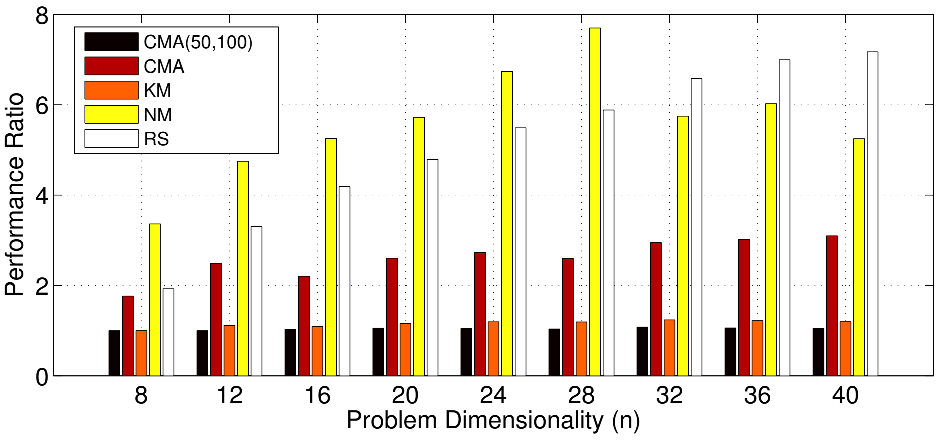

Fig. 1. Performance results (mean best fitness) for the Iris (I) dataset problems, as a ratio with the globally optimal value (e.g. a value of 2.5 means the average best solution found by an algorithm was 2.5 times the value of the global optimum.

Table 2. Results for clustering problems. Problem instances are defined by a dataset (D), i.e. Iris (I), Ruspini (R) or German towns (G), together with the problem dimensionality (n), where $n = dk$. Shown are mean and standard deviations over 50 trials of each algorithm. The mean and average number of function evaluations ($\#f$) is also shown for each algorithm. A result is in bold if one or more of the 50 trials located the global optimum (f^*) to at least 15 significant figures.

D	n	NM	NM $\#f$	RS	KM	KM $\#f$
I	8	**5.1208e02(2.5e02)**	1.0938e03(5.8e02)	2.93656e02(4.1e01)	**1.52348e02(0.0e00)**	4.66e00(1.4e00)
	12	3.7454e02(2.6e02)	1.6421e03(6.1e02)	2.60478e02(3.0e01)	**8.79682e01(2.3e01)**	6.94e00(2.9e00)
	16	3.0047e02(2.4e02)	2.0770e03(5.5e02)	2.39601e02(2.8e01)	**6.24359e01(6.9e00)**	8.2e00(3.7e00)
	20	2.6579e02(2.2e02)	2.6568e03(8.8e02)	2.2239e02(2.7e01)	**5.37562e01(8.8e00)**	9.18e00(3.6e00)
	24	2.6285e02(2.3e02)	3.0913e03(1.3e03)	2.1431e02(1.9e01)	4.66559e01(8.7e00)	7.86e00(2.9e00)
	28	2.6404e02(2.2e02)	3.8847e03(2.8e03)	2.0180e02(2.4e01)	4.09041e01(5.5e00)	8.82e00(3.4e00)
	32	1.7233e02(1.3e02)	4.3597e03(1.4e03)	1.9722e02(1.8e01)	3.70879e01(7.5e00)	8.34e00(2.8e00)
	36	1.6729e02(1.1e02)	5.0430e03(2.8e03)	1.9432e02(1.9e01)	3.39236e01(3.8e00)	7.78e00(2.1e00)
	40	1.3556e02(3.2e01)	5.8026e03(2.3e03)	1.8527e02(1.8e01)	3.09606e01(10.3e01)	8.52e00(3.0e00)
R	4	**8.9338e04(0.0e00)**	5.03e02(2.5e01)	9.01376e04(4.0e02)	**9.67212e04(2.3e04)**	3.82e00(1.5e00)
	6	6.6428e04(2.1e04)	8.004e02(1.3e02)	5.37281e04(1.0e03)	**5.11096e04(4.6e01)**	3.70e00(1.5e00)
	8	**3.5846e04(2.1e04)**	1.275e03(3.3e02)	1.99433e04(2.3e03)	**2.83654e04(1.8e04)**	3.99e00(1.6e00)
	10	3.5366e04(2.1e04)	1.9712e03(1.3e03)	1.65256e04(1.6e03)	**1.86163e04(1.5e04)**	4.45e00(1.6e00)
	12	2.7332e04(2.0e04)	4.203e03(2.8e03)	1.50499e04(1.0e03)	**1.46650e04(1.2e04)**	4.66e00(1.7e00)
	14	2.6147e04(2.0e04)	7.4512e03(2.8e03)	1.36176e04(1.0e03)	**1.07800e04(7.7e03)**	5.08e00(1.7e00)
	16	2.5967e04(2.1e04)	7.5234e03(4.4e03)	1.24834e04(1.0e03)	**9.58363e03(6.8e03)**	5.00e00(1.7e00)
	18	2.4995e04(2.1e04)	1.1427e04(3.9e03)	1.15749e04(7.4e02)	8.22721e03(5.1e03)	5.01e00(1.7e00)
	20	1.0728e04(1.2e03)	5.9614e03(2.1e03)	1.09593e04(7.1e02)	6.97757e03(3.9e03)	5.30e00(1.7e00)
G	6	1.5526e12(8.7e11)	9.908e02(4.1e02)	1.08704e12(1.6e11)	**6.51273e11(1.6e10)**	5.95e00(1.4e00)
	9	**4.1772e11(1.7e11)**	2.8568e03(1.4e03)	9.12653e11(1.3e11)	**3.63195e11(3.7e09)**	9.07e00(1.5e00)
	12	5.7125e11(1.7e11)	2.1122e03(6.1e02)	8.50900e11(9.5e10)	**2.73230e11(3.1e10)**	1.51e01(2.8e00)
	15	4.9316e11(1.5e11)	3.0132e03(1.8e03)	7.83531e11(8.3e10)	**2.49510e11(4.4e10)**	1.55e01(3.2e00)
	18	3.3045e11(1.8e11)	7.8998e03(7.5e03)	7.50905e11(9.0e10)	2.29057e11(6.0e10)	1.53e01(5.3e00)
	21	3.2013e11(3.5e10)	5.0868e03(4.4e02)	6.66544e11(8.9e10)	2.09148e11(8.1e10)	1.43e01(4.2e00)
	24	4.5023e11(1.8e11)	4.2644e03(1.4e03)	6.67006e11(8.7e10)	1.79267e11(9.8e10)	1.39e01(4.3e00)
	27	2.8713e11(2.1e11)	7.598e03(4.1e03)	6.50674e11(8.4e10)	1.35754e11(1.1e11)	1.33e01(3.6e00)
	30	2.7033e11(9.7e10)	6.4218e03(1.5e03)	6.34447e11(8.7e10)	1.10532e11(1.0e11)	1.32e01(4.0e00)

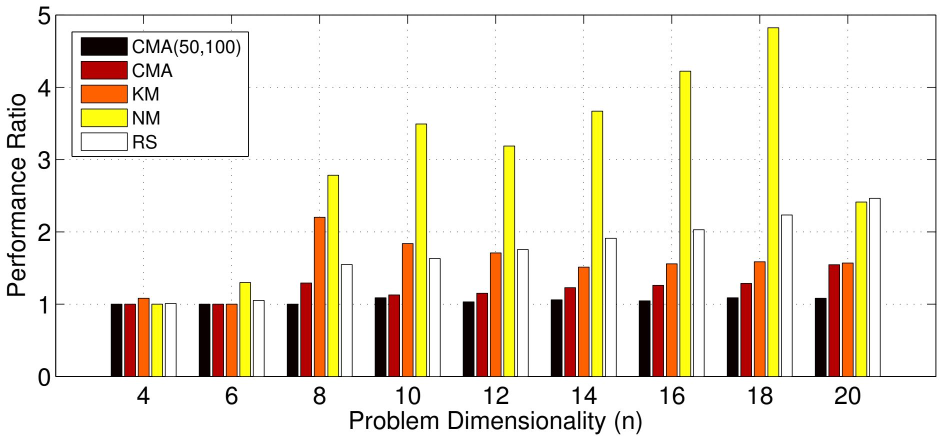

Fig. 2. Performance results (mean best fitness) for the Ruspini (R) dataset problems, as a ratio with the globally optimal value.

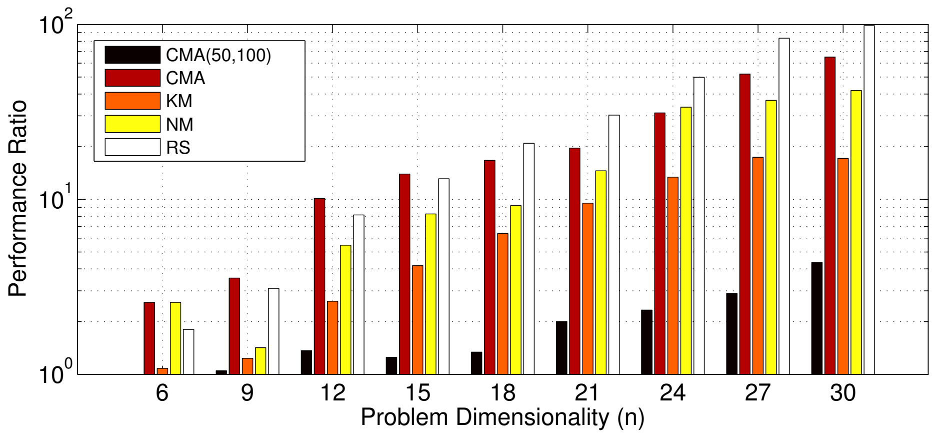

Fig. 3. Performance results (mean best fitness) for the German towns (G) dataset problems, as a ratio with the globally optimal value.

The trends as problem dimensionality increases give some indication of the scaling behaviour of the performance of each algorithm. As expected, random search steadily increases (given a fixed budget of function evaluations). Some individual problem instances also appear to be particularly challenging for the algorithms tested. For example, the 8D Ruspini problem appears more difficult for KM and CMA, than both the 6D *and* the 10D Ruspini problem instances (Fig.2). Note also that the German towns problems lead to poor performance ratios for the algorithms compared to the other datasets (in Fig.3 the y-axis is on a log scale).

The comparative results in Figs 1-3 give a general indication of performance, but it is important to note that average fitness values are a relatively gross summary of the results and may hide important details. For example, the average performance of CMA on the 12D and 15D German towns problems (Fig.3) is slightly worse than random search! However, Table 1 shows that the standard deviation of CMA results on these problems is relatively large. Further investigation of these results revealed that many trials found solutions much better than the average, but a number of other trials converged to a poor solution considerably worse. Hence, the average is a poor summary of such results.

6.1 Comparison with Previous Results

Table 3 shows the results reported by previous papers for one of the problem instances tested, Iris with $k = 3$. A variety of algorithms have been tested on this problem. Comparing these results with those from above, the most striking thing is that *all* of these reported results are relatively poor. The average objective function values are far from optimal and are significantly outperformed by CMA(50,100). These results tend to be based on fewer function evaluations, but the papers do not seem to be targetting a "low budget" scenario, but rather evaluating the potential of the algorithms. Another significant anomaly is the differences between the k-means (KM, SBKM) results reported in these papers (average values between 97 and 101) and the result obtained in this paper for KM on this problem (8.79682e01(2.3e01)). There may be a difference in the initialization technique used which is not mentioned in all papers. In any case, there are clearly unresolved questions here, demonstrating the need for the development of standard specifications and experimental practice when evaluating black-box optimization algorithms.

Table 3. Previous results for the Iris (k=3, n=12) problem. A question mark (?) means that it is not clear from the paper what value was used in the experiments.

Reference	Algorithm	f_{ave}	Evals
[8]	GA	97.10077 (5 times!)	10^4
[8]	KM (best trial of 5)	97.204574	
[14]	KM	98.1872	?
[14]	Fuzzy c-means	96.9280	?
[14]	AKPSO	96.7551	?
[6]	KM	99.84	10^4
[6]	Shelokar ACO	97.78	10^4
[6]	ACOC	97.22	10^4
[5]	HBM	96.95316	11214
[5]	ACO	97.171546	10998
[5]	GA	125.197025	38128
[5]	TS	97.868008	20201
[5]	SA	97.134625	29103
[12]	SBKM	101.3672	3e04(?)
[12]	GAPS	97.3868	3e04(?)
[12]	VGAPS	96.2022	3e04(?)
[12]	PSO	96.0176	3e04(?)
[12]	ACO	99.9176	3e04(?)
[12]	SA	101.4574	3e04(?)
[12]	BNMR	95.0927	3e04(?)

7 Summary

This paper has examined sum of squares clustering problems as a source of real-world benchmark problems for the evaluation and comparison of black-box optimization algorithms. It was shown that clustering problems have many useful properties for benchmarking. To facilitate better comparisons of algorithms and experimental results, a specification was provided for clustering problems and a web repository has been created. Experimental results were presented on a set of 27 clustering problems and some comparisons made with existing results in the literature. It is intended that future work will build on and add to the problems specified here, with additional datasets. Also, future research should be able to make better use of published experimental results on clustering problems.

References

1. C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases. Retrieved from http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.
2. J. Brimberg, P. Hansen, N. Mladenovic, and E. D. Taillard. Improvements and comparison of heuristics for solving the uncapacitated multisource Weber problem. *Operations Research*, 48(3):444–460, 2000.
3. Dong-Xia Chang, Xian-Da Zhang, and Chang-Wen Zheng. A genetic algorithm with gene rearrangement for k-means clustering. *Pattern Recognition*, 42(7):1210–1222, 2009.
4. O Du Merle, P Hansen, B Jaumard, and N Mladenovic. An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, 21(4):1485–1505, 2000.
5. Mohammad Fathian, Babak Amiri, and Ali Maroosi. Application of honey-bee mating optimization algorithm on clustering. *Applied Mathematics and Computation*, 190(2):1502–1513, 2007.
6. Yucheng Kao and Kevin Cheng. An aco-based clustering algorithm. In *Ant Colony Optimization and Swarm Intelligence*, pages 340–347. Springer, 2006.
7. Ruochen Liu, Zhengchun Shen, Licheng Jiao, and Wei Zhang. Immunodomaince based clonal selection clustering algorithm. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7, 2010.
8. Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465, 2000.
9. S. Salhi and M. D. H. Gamal. A genetic algorithm based approach for the uncapacitated continuous location-allocation problem. *Annals of Operations Research*, 123:230–222, 2003.
10. PS Shelokar, Valadi K Jayaraman, and Bhaskar D Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, 2004.
11. D. Steinley. K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59:1–34, 2006.
12. Mohammad Taherdangkoo, Mohammad Hossein Shirzadi, Mehran Yazdi, and Mohammad Hadi Bagheri. A robust clustering method based on blind, naked mole-rats (bnmr) algorithm. *Swarm and Evolutionary Computation*, 10:1–11, 2013.
13. R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.
14. Fun Ye and Ching-Yi Chen. Alternative kpso-clustering algorithm. *Tamkang Journal of science and Engineering*, 8(2):165, 2005.