Chameleon 2: An Improved Graph-Based Clustering Algorithm

TOMAS BARTON, Czech Technical University in Prague, Institute of Molecular Genetics ASCR
TOMAS BRUNA and PAVEL KORDIK, Czech Technical University in Prague

Traditional clustering algorithms fail to produce human-like results when confronted with data of variable density, complex distributions, or in the presence of noise. We propose an improved graph-based clustering algorithm called Chameleon 2, which overcomes several drawbacks of state-of-the-art clustering approaches. We modified the internal cluster quality measure and added an extra step to ensure algorithm robustness. Our results reveal a significant positive impact on the clustering quality measured by Normalized Mutual Information on 32 artificial datasets used in the clustering literature. This significant improvement is also confirmed on real-world datasets.

The performance of clustering algorithms such as DBSCAN is extremely parameter sensitive, and exhaustive manual parameter tuning is necessary to obtain a meaningful result. All hierarchical clustering methods are very sensitive to cutoff selection, and a human expert is often required to find the true cutoff for each clustering result. We present an automated cutoff selection method that enables the Chameleon 2 algorithm to generate high-quality clustering in autonomous mode.

CCS Concepts: • **Information systems** → **Clustering**; **Clustering and classification**; *Nearest-neighbor search*; • **Computing methodologies** → **Cluster analysis**; **Machine learning algorithms**; *Anomaly detection*; • **Theory of computation** → *Unsupervised learning and clustering*; • **Mathematics of computing** → Hypergraphs; Graph algorithms;

Additional Key Words and Phrases: Cluster analysis, clustering, graph clustering, pattern recognition

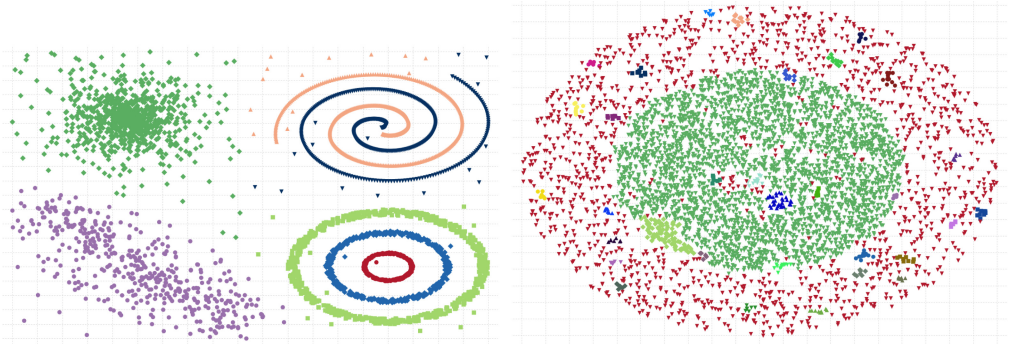
ACM Reference format:
Tomas Barton, Tomas Bruna, and Pavel Kordik. 2019. Chameleon 2: An Improved Graph-Based Clustering Algorithm. *ACM Trans. Knowl. Discov. Data* 13, 1, Article 10 (January 2019), 27 pages.
https://doi.org/10.1145/3299876

1 INTRODUCTION

Clustering is a technique that attempts to group similar objects into clusters and dissimilar objects into different clusters (Kaufman and Rousseeuw 1990). There is no general consensus to what exactly a cluster is. In fact, it is generally acknowledged that the problem is ill-defined (Caruana et al.

This research was partially supported by CTU Grant SGS17/210/OHK3/3T/18 Modern data-mining methods for advanced extraction of information from data; NPU I (LO1220) of the Ministry of Education, Youth and Sports of Czech Republic; GA18-18080S of the Grant Agency of the Czech Republic; and ERDF Grant Upgrade of National Infrastructure for Chemical Biology (No. CZ.02.1.01/0.0/0.0/16_013/0001799).
Authors' addresses: T. Barton and P. Kordik, Department of Applied Mathematics, Faculty of Information Technology, Czech Technical University in Prague, Thakurova 9, Prague 6, 160 00, Czech Republic; emails: {tomas.barton, kordikp}@fit.cvut.cz; T. Bruna, School of Biology, Georgia Institute of Technology, North Avenue, Atlanta, GA 30332, USA; email: bruna.tomas@gatech.edu.

1556-4681/2019/01-ART10 $15.00
https://doi.org/10.1145/3299876



(a) Impossible dataset (b) DBSCAN on disk-in-disk dataset

Fig. 1. (a) A Chameleon 2 result on a toy dataset, inspired by Jain et al., who suggest that it cannot be solved by a clustering algorithm. Our proposed algorithm gives a solution very close to human judgment. (b) DBSCAN clustering of *dense-disk-5k* dataset with best configuration found ($minPts = 5$, $\varepsilon = 0.28$) includes many non-sense clusters, because DBSCAN cannot handle different densities of clusters.

2006). Various algorithms use slightly different definitions of a cluster, e.g., based on the distance to the closest cluster center or the density of objects in an object's neighborhood. Unlike supervised learning where labeled data are used to train a model, which is afterwards used to classify unseen data, clustering is an unsupervised problem. That is, assign data to *a priori* unknown number of groups based on their similarity. Typically more than one solution exists. Clustering is more challenging than classification (Jain 2010), with applications in many fields, including data-mining, machine learning, marketing, biology, chemistry, astrology, psychology, and spatial database technology.

Due to these varied domains of application, most respected authors define clustering in a vague way (Jain and Dubes 1988; Everitt 1993), leaving room for several interpretations. Generally, clustering algorithms are designed to group objects in the same way as a human observer would. Nonetheless the ultimate goal is to detect structures in higher dimensions than humans are capable of recognizing. How such methods should be evaluated is yet another problem. In this contribution, we mostly focus on patterns that can be easily distinguished by humans.

As demonstrated by Jain (2010) a toy problem easily solvable by a human seems to be impossible for a traditional clustering algorithm. Our proposed algorithm attempts to solve this problem in the way a human would (Figure 1(a) shows the actual clustering result using our method). On this problem many algorithms fail to produce human-competitive clustering results. So far hundreds of algorithms have been proposed (Jain 2010; Aggarwal and Reddy 2014). Some assign items to exactly one cluster, while others allow fuzzy assignment to many clusters. There are methods based on cluster prototypes, mixture models, graph structures, neighborhood density, and grid-based models. The most common approach to representing information contained in base clustering is to use similarity matrices.

The well-known k-means algorithm, which is usually referred to as Lloyd's algorithm, was proposed in 1957, although published only in 1982 (Lloyd 1982). The algorithm itself was independently discovered in several fields (Ball and Hall 1965; MacQueen 1967). Since then many algorithms have been proposed, but k-means remains quite popular due to its simplicity and nearly linear computational complexity. The fact that k-means is still preferred in many cases also emphasizes the difficulty of designing a general purpose clustering algorithm. The main disadvantage of prototype approaches is their assignment of data points to the nearest centroid; such methods fail to discover non-spherical clusters when Euclidean distance is used (Jain 2010).

Chameleon (Karypis et al. 1999b) is a graph-based clustering algorithm, which attempts to overcome the limitations of traditional clustering approaches. It works by combining the proximity and connectivity of data points as well as taking into account internal cluster properties during the merging phase. Unlike other algorithms, Chameleon produces as human-like results as possible.

The original Chameleon often manages to identify key clusters, but produces extra noisy clusters and moreover it is hard to configure. In order to stabilize the results and simplify configuration of the algorithm, we propose an improved version – called Chameleon 2 – that is capable of finding complex structures in various datasets with a minimal error rate and without the necessity of fine parameter tuning. In this article, we describe the modifications made and compare the altered version with the original version of Chameleon, as well as with other clustering algorithms. According to our benchmarks, the new Chameleon algorithm outperforms the original one on all datasets and almost always provides better results than other commonly used algorithms.

The article is organized as follows. Section 2 presents related methods. Section 3 describes the original Chameleon algorithm, and in Section 4, we present our approach. Furthermore, in Section 5, we focus on the automatic selection of high quality clustering from hierarchical results. Section 6 then compares Chameleon 2 to other approaches against an extensive benchmark taken from artificial datasets used in the clustering literature, and the final section concludes this article with a summary of our contributions.

2 RELATED WORK

Many clustering algorithms have been proposed in the past; in this section, we include related approaches that lead to high-quality clustering. Some algorithms are density based (Lance and Williams 1967; Guha et al. 1998) while others use graph representation (Zhao and Karypis 2001) or fit models to data distributions (Dempster et al. 1977; McLachlan and Basford 1988).

Hierarchical agglomerative clustering (HC) is one of the oldest clustering algorithms that uses a bottom-up approach to merge the closest items and produce a hierarchical data structure. There are several methods for computing cluster similarity, to discover arbitrarily shaped clusters, the most suitable method is the *single-linkage* method that computes the similarity of clusters by their closest items; however, this method cannot deal with outliers or noise (Jain et al. 2004).

One of first approaches that measures similarity by the number of shared neighbors was proposed by Jarvis and Patrick (1973). The later DBSCAN (Ester et al. 1996) takes a very similar approach; the algorithm is capable of discovering arbitrarily shaped clusters if the cluster density can be determined beforehand and each cluster has a uniform distribution. A cluster is defined as a maximum set of density-connected points, where every core point in a cluster must have at least a minimum number of points (*minPts*) within a given radius (ε). All points within one cluster can be reached by traversing a path of density-connected points. The algorithm itself can be relatively fast, however in order to configure its parameters properly, prior knowledge of the dataset or landmarking by k-NN algorithm is required. The main disadvantage of DBSCAN is its sensitivity to parameter values: even a small modification of the ε parameter could cause all data points to be assigned to a single cluster. Moreover, the algorithm will fail on datasets with non-uniform distribution (as illustrated in Figure 1(b)). The original article incorrectly states $O(n \log(n))$ time complexity that could be achieved only in 2D space. For $d \geq 3$ the problem requires $\Omega(n^{4/3})$ time to solve as Gan and Tao (2015) showed. The HDBSCAN (McInnes et al. 2017) algorithm decreases the parameter sensitivity of the original algorithm by performing DBSCAN over varying epsilon values and integrates the result to find a clustering that gives the best stability over epsilon. In our experiments, we use the original version of the algorithm and perform an exhaustive search over its parameters.

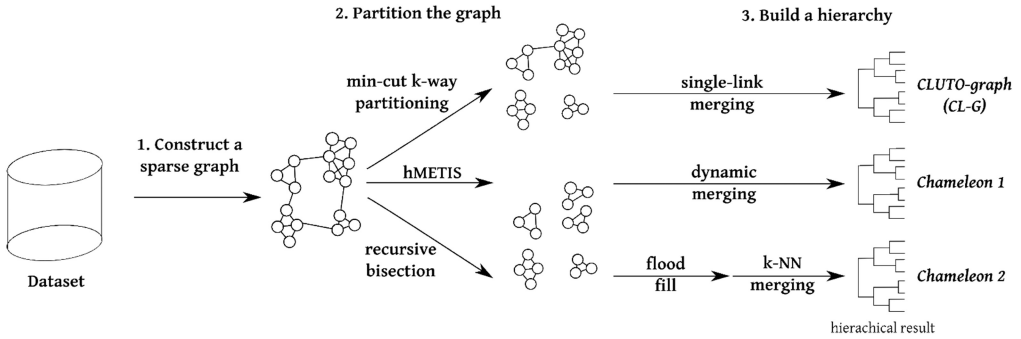


Fig. 2. An overview of Chameleon family approaches. All these algorithms could be considered hybrid in terms of the traditional division between agglomerative and partitioning algorithms. These algorithms combine the partitioning phase with the agglomerative phase while producing a hierarchical result.

CURE[1] (Guha et al. 1998) is a clustering algorithm that uses a variety of different techniques to address various drawbacks of the HC method. Each cluster is represented by multiple points, issues with scaling are addressed by data subsampling. CURE is more robust to outliers than a HC algorithm, and it also manages to identify clusters of a non-spherical shape and of widely varied sizes.

The algorithm called Chameleon by Karypis et al. (1999b) proposes combining interconnectivity and closeness of clusters. According to the authors the algorithm yields accurate clustering results on highly variable data. In order to identify clusters, Chameleon uses a three-step approach. The first two steps aim to partition the data into many small subclusters and the last repeatedly merges these subclusters into the final result. The whole process is illustrated in Figure 2. Chameleon has received much attention in the literature, however most authors mention the algorithm as an interesting graph-based clustering approach but do not investigate the further properties of the algorithm. Shatovska et al. (2007) was unable to fully reproduce the results obtained by Karypis et al. (1999b).

Zhao and Karypis (2001) applied a Chameleon-like algorithm in the area of document clustering (high-dimensional data). Their first phase is the same as in the Chameleon approach: A graph is constructed with the k-NN. Afterwards the graph is split into k clusters using a min-cut graph partitioning algorithm. Finally a hierarchical clustering is produced using a *single-link* algorithm for partitioned clusters from the previous phase. The algorithm is implemented in the CLUTO[2] package (Karypis 2002a), and in the rest of this article we will refer to this algorithm as CL-G.

Jain et al. (2004) constructed a taxonomy of existing clustering algorithms and reported that the Chameleon algorithm family provides very similar results even with various configurations, but that they are very different from those produced by other algorithms. They used the CLUTO implementation in their experiments, thus the algorithm used was not Chameleon but CL-G. Their results were similar because all variants of Chameleon used Cosine distance, while other approaches used Euclidean distance.

Newman and Girvan (2004) proposed a hierarchical divisive algorithm for community detection in large graphs that laid the foundations for many follow up works. However, this approach is

[1]Clustering Using REpresentatives.
[2]CLUTO (CLUstering TOolkit) supports other partitioning methods that do not work with graph representation, but partition data based on an objective function using cosine similarity. Such methods are not comparable in our low-dimensional benchmark.

suitable for a different set of problems, in which a predefined network is given as an input of the algorithm.

Chinese Whispers (CW) from Biemann (2006) is yet another graph-based clustering algorithm. The approach is relatively simple, and it is intended for clustering large graphs. Like Chameleon, the algorithm can be initialized using k-NN and afterwards assigns items to clusters based on the majority label in their neighborhood (whisper spreading). The algorithm stops after a given number of iterations or when no change is made during a single iteration.

Affinity Propagation (AP) (Frey and Dueck 2007) identifies cluster prototypes (exemplars) by sending messages between data points and reports their neighborhoods as clusters. All data points are considered as possible prototypes. The final set of cluster prototypes is determined by voting.

Handl and Knowles (2007) designed a multi-objective evolutionary clustering algorithm called MOCK that selects clusterings based on objectives called *connectivity* and *deviation* that are computed in a similar way to Chameleon's metrics. In both approaches, connectivity (resp. interconnectivity) metrics are based on k-NN computation.

More recently Rodriguez and Laio (2014) presented a density-based approach based on the idea that clusters are surrounded by lower density regions. The algorithm requires a distance matrix as an input, and Gaussian kernels are then used to estimate density.

Most recently Barton et al. (2016) described a multi-objective version of Chameleon. Instead of using a linear combination of two objectives, a Pareto-optimal pair is selected during the merging phase.

There are a number of approaches for extracting clusters from hierarchical structures, such as the gap statistic (Tibshirani et al. 2001). Most such methods rely on internal cluster evaluation metrics that might be data-specific and in many cases computationally expensive.

3 ORIGINAL CHAMELEON ALGORITHM

The Chameleon algorithm operates on a graph representation of data. The first step is to construct a graph based on k nearest neighbors. A data point, represented by a node in graph, is connected with its k closest neighbors by an edge that is given weight equal to the inversion of their distance. If a graph is provided, we can skip this step.

The second step *partitions* the previously created graph. The goal of partitioning is to produce equal-sized partitions and minimize the number of edges cut. In this manner, many small clusters are formed with a few highly connected nodes in each cluster. For partitioning, Karypis et al. use their own hyper-graph partitioning algorithm called hMETIS, which is a multi-level partitioning algorithm working with a coarsened version of the graph. Coarsening is non-deterministic, thus for each run we might obtain a slightly different result (for further details see Karypis et al. (1999a) and Karypis and Kumar (1999)).

The final and most important step *merges* the partitioned clusters using Chameleon's dynamic modeling framework, starting with the most similar pair of clusters. This procedure is common for all agglomerative algorithms: The most similar pair of all possible pairs of clusters is selected and merged to form a larger cluster. There are $n \cdot (n-1)/2$ such pairs, thus the time and memory complexity is $O(n^2)$, however thanks to the previous partitioning we are merging clusters containing multiple items, so the number of merges is far smaller than in traditional hierarchical clustering.[3] There are many ways of computing similarities between clusters, e.g., based solely on the distance between two points (*single-link*, *complete-link*, the distance from a cluster's centroid, etc.) or

[3]The actual speed-up depends on the number of neighbors in the initial graph (parameter k) and the separation of items in the dataset. If the dataset contains many isolated points, each of those points will form an individual cluster after partitioning, which has then to be compared to all other clusters.

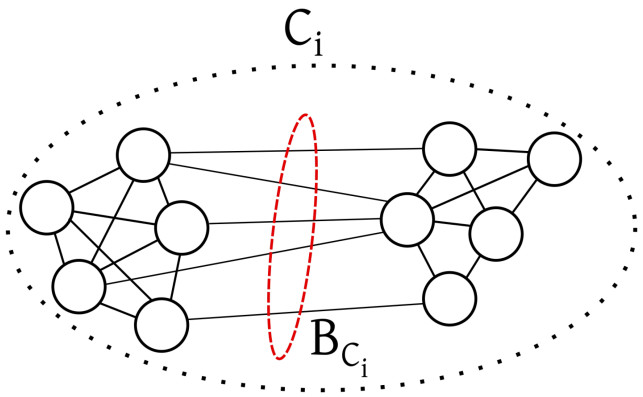


Fig. 3. All nodes in this graph belong to a single cluster C_i, in order to compute bisection $\phi(C_i)$, we are looking for a set of edges B_{C_i} (marked with dashed line) whose removal would split the cluster into two roughly equal parts.

using multiple points (e.g., *average-link*). Chameleon's approach belongs to the latter category, but in addition to computing distances between clusters it also accounts for intra-cluster distances. This idea is not entirely new; there are many examples of between cluster validation metrics (such as Davies and Bouldin (1979) or Caliński and Harabasz (1974)), which compute the ratio between cluster compactness and cluster separation. Nonetheless, in Chameleon both similarity metrics are computed during the merging phase, while in other approaches evaluation metrics are typically used for external result evaluation only.

3.1 Chameleon's Internal Metrics

During merging, Chameleon tries to merge clusters of similar densities. Density estimation relies on graph bisection, i.e., the division of a cluster (graph) into two roughly equal parts by cutting a minimal number of edges. The quality of the bisection algorithm used has major impact on the merging order, and thus influences the final clustering result.

We introduce a notation used to define the similarity metrics, in which each cluster is represented as a graph. $B_{C_i} = \text{bisect}(C_i)$ is a set of edges selected by a bisection algorithm. $w(e)$ is the weight (typically Euclidean distance inversion) of a given edge that was computed by the k-NN algorithm:

$$\phi(C_i) = \sum_{e \in B_{C_i}} w(e), \tag{1}$$

$$\bar{\phi}(C_i) = \frac{1}{|B_{C_i}|}\phi(C_i), \tag{2}$$

where $\phi(C_i)$ represents the sum of the edges' weights in B_{C_i}, and the average edge weight is noted as $\bar{\phi}(C_i)$ (an example is shown in Figure 3).

The between-cluster metrics are defined in a similar manner: the sum of the between-cluster edges' weights $s(C_i, C_j)$ and the average weight of all edges between clusters $\bar{s}(C_i, C_j)$, except that it is not necessary to compute a bisection in this case, because we already have two clusters.

The first measure employed by Chameleon is called *relative closeness*:

$$R_{CL}(C_i, C_j) = \frac{\bar{s}(C_i, C_j)}{\frac{|E_{C_i}|}{|E_{C_i}|+|E_{C_j}|}\bar{\phi}(C_i) + \frac{|E_{C_j}|}{|E_{C_i}|+|E_{C_j}|}\bar{\phi}(C_j)}, \tag{3}$$

where $|E_{C_i}|$ and $|E_{C_j}|$ represent the number of edges inside clusters C_i and C_j, respectively. The numerator is similar to the *average link* method, but only distances between nodes connected by an edge are included. The denominator uses average inter-cluster distances, which are normalized by the relative number of edges in each cluster (again an edge must be present between pairs in a cluster in order to be included in this denominator). Using the same notation we define *relative inter-connectivity*:

$$R_{IC}(C_i, C_j) = \frac{s(C_i, C_j)}{\frac{\phi(C_i)+\phi(C_j)}{2}}. \tag{4}$$

High values of inter-connectivity indicate that clusters C_i and C_j should be merged. The summed weight of the edges between clusters is compared to the sum of the weights after cluster bisection. This normalization drives Chameleon to merge clusters with similar densities.

Finally, the *Ch1* similarity function between two clusters (C_i and C_j) is computed as Karypis et al. (1999b):

$$S_{\text{Ch1}}(C_i, C_j) = R_{CL}(C_i, C_j)^{\alpha} \cdot R_{IC}(C_i, C_j)^{\beta}, \tag{5}$$

where α, β are user specified parameters that control the tradeoff between either relative closeness (compact clusters) or relative inter-connectivity (well-separated clusters). Default values are $\alpha = 2.0, \beta = 1.0$.

3.2 Chameleon Drawbacks

Although Chameleon is considered to be one of the best hierarchical clustering algorithms (Aggarwal and Reddy 2014), our experiments reveal several deficiencies in the final clustering. The main drawback of the algorithm is its inability to handle singleton clusters (clusters containing one or few items). Moreover, unlike DBSCAN and CURE, Chameleon does not handle noise at all.

Figure 5(a) illustrates a typical problem with many disconnected clusters in low density regions after hMETIS partitioning. The solution is contaminated with many singleton clusters that should be assigned to neighboring clusters instead of forming individual clusters. This is not solely an issue of partitioning, but also a sign that the similarity metric does not successfully capture the notion of a point's neighborhood, because close points are not merged together. Most problematic is the R_{IC} definition. The sum of bisected edges penalizes clusters that can be easily divided into two parts.

Another problem is the absence of an algorithm for the extraction of high-quality partitioning from hierarchical structures created by the merging process. This problem is common to all agglomerative algorithms.

Implementation of the Chameleon algorithm as it is described in Karypis et al. (1999b) was not available at the time of writing. The most similar is the CL-G algorithm included in CLUTO package (Karypis 2002a) (see Figure 2), however it shares only the first phase with Chameleon and the rest of the algorithm is different.

4 INTRODUCING CHAMELEON 2 ALGORITHM

Several experiments with the Chameleon algorithm have shown that configuring the algorithm is complicated. Besides parameters k, α, β, several hMETIS parameters also need to be adjusted for the specific dataset. Even with the best possible configuration the algorithm still produces noisy clustering. Therefore, we decided to design an improved version of the Chameleon algorithm (Ch1), which we refer as Chameleon 2 (Ch2). The individual changes we describe below significantly improve the algorithm's performance as we demonstrate in Section 6.

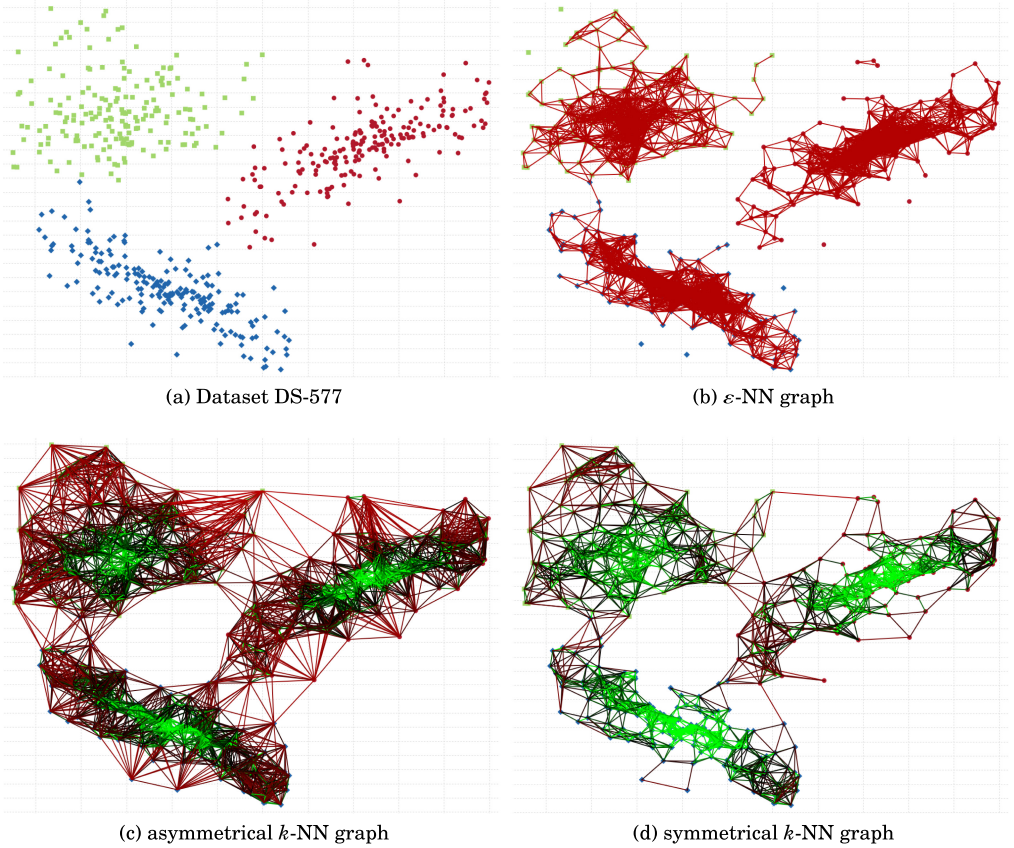


Fig. 4. (a) Dataset *DS-577* with 577 data points that contains 3 true clusters. (b) Graph constructed using range queries where edge is added when distance between two points is lower than ε ($\varepsilon = 0.33$) on dataset *DS-577*. (c) Each data point is connected by an edge with its $k = 18$ neighbors. Green edges are shorter than red edges. (d) Symmetrical k-NN – an edge is added only if both nodes list the other node between its 18 neighbors.

4.1 Graph Initialization

A multi-dimensional dataset can be converted into a graph structure by many methods. Figure 4 shows conversion to graph representation on dataset *DS-577*. A naive approach would be applying ε-NN (Figure 4(b)), which requires a fine-tuned ε value and often the resulting graph contains several components that need to be specially treated. While Chameleon 1 relies on asymmetrical k-NN (Figure 4(c)), Chameleon 2 starts with symmetrical k-NN (Figure 4(d)), which contains substantially fewer edges. In asymmetrical k-NN, each item is connected by an edge to its k nearest neighbors. Symmetrical k-NN constructs a graph where an edge connects two nodes only if both nodes belong to the k nearest neighbors of the other node. This eliminates many between cluster connections and leads to better clustering results in the end.

In order to obtain reasonable clustering an appropriate k-value needs to be set. The algorithm works best if a connected graph is provided. Brito et al. (1997) proved that for large enough n and some constant c setting $k = c \cdot \ln(n)$ a connected graph is guaranteed. Throughout our experiments parameter k is computed as $2 \cdot \ln(n)$, where n is the input data size.

4.2 Graph Partitioning

Chameleon 1 uses fast hyper-graph partitioning (hMETIS), nonetheless the algorithm is non-deterministic. Moreover, the source code is not available. This makes further improvements rather difficult.[4] Therefore, we implemented an alternative partitioning method based on the recursive Fiduccia–Mattheyses bisection (Fiduccia and Mattheyses 1982). The *recursive bisection* procedure is simple and requires one parameter $p_{\max}$, defining the maximum number of objects in a single partition. The Fiduccia–Mattheyses bisection is repeated until all clusters contain no more than $p_{\max}$ objects.

4.3 Partition Refinement

There is a significant problem with both hMETIS and recursive bisection partitioning methods. In their effort to create a balanced partitioning with the fewest possible edges cut, the algorithms can sometimes produce clusters that include items that are very far apart and not connected by any edges. Such clusters cannot be fixed in the merging phase and seriously worsen the final result. This problem is illustrated in Figure 6.

To remedy this situation we apply a method called *flood fill* which recursively finds connected components in the partitioned graph. If a cluster is formed by several disconnected components, flood fill splits it into separate, connected clusters. The flood fill process is shown in Algorithm 1. An example of Chameleon 2 run with the flood fill refinement is shown in Figure 5(b) and (c).

ALGORITHM 1: Flood Fill

Input: Partitioned k-NN *graph*.
Output: Connected clusters.

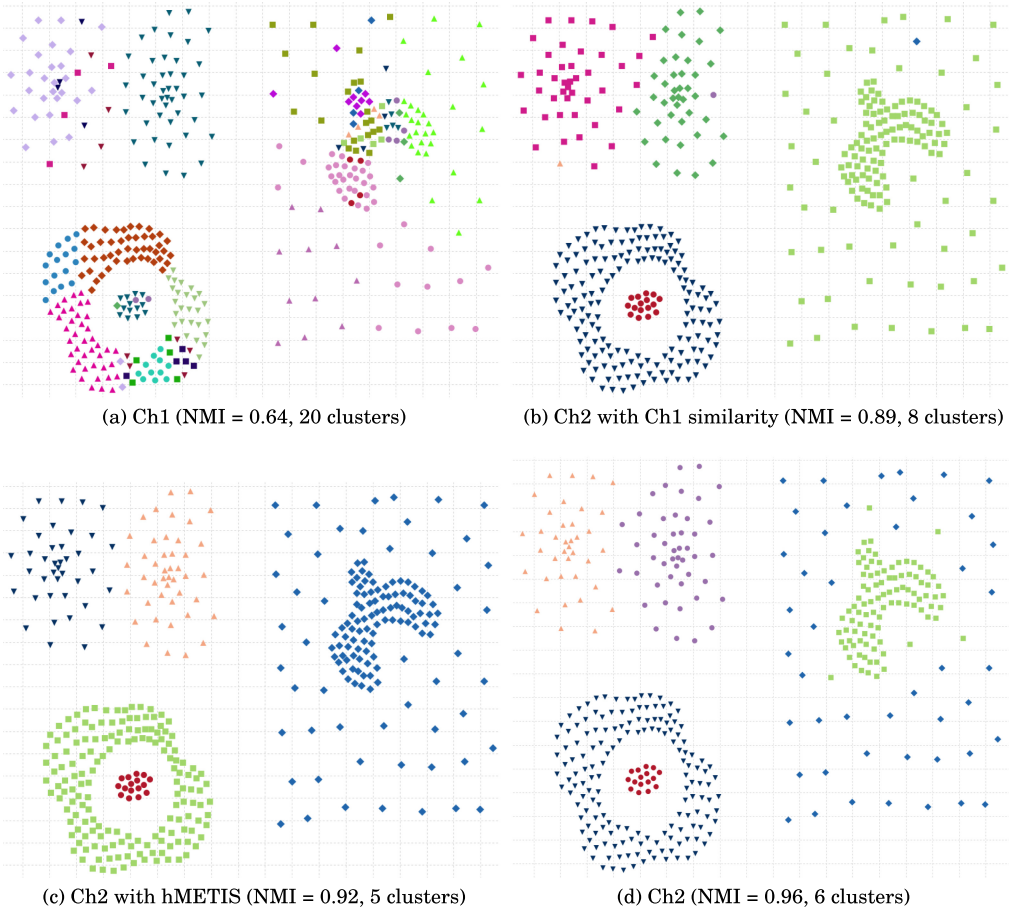
```
find and mark connected subgraphs
Function FloodFill(graph)
    while not marked all nodes in graph do
        foreach node in graph do
            if not marked(node) then
                cluster = NewCluster()
                MarkConnectedSubgraph(graph, node, cluster)

recursively add all connected nodes to the same cluster
Function MarkConnectedSubgraph(graph, Node, cluster)
    Mark (node, cluster)
    foreach neighbor in neighbors(node) do
        if not marked(neighbor) then
            MarkConnectedSubgraph(graph, neighbor, cluster)
```

4.4 Merging

Unlike traditional hierarchical algorithms Chameleon 2 stores nearest neighboring clusters for each cluster, as we are merging clusters that are close to each other. The distance between clusters is based on metrics that are introduced in the next section. Instead of computing $O(n^2)$ distances

[4]The hMETIS license allows usage for educational, research, and benchmarking purposes by non-profit institutions and US government agencies only.



(a) Ch1 (NMI = 0.64, 20 clusters)

(b) Ch2 with Ch1 similarity (NMI = 0.89, 8 clusters)

(c) Ch2 with hMETIS (NMI = 0.92, 5 clusters)

(d) Ch2 (NMI = 0.96, 6 clusters)

Fig. 5. Chameleon 1 and 2 (using $k = 10$, $\alpha = 2$, $\beta = 1$ and hMETIS partitioning) on the *compound* dataset. For evaluation Normalized Mutual Information (NMI) is used, this metric is further explained in Section 6.1 Methodology. (a) Default configuration with standard similarity; the overall quality of clustering is unexpectedly poor. There are the following two reasons for this: first, after hMETIS partitioning many singleton clusters are left, because meaningful similarity computing is complicated for small groups of data points. Second, the merge function does not provide locally consistent ordering, which results in many noisy clusters. (b) Nearly Chameleon 2 algorithm, this uses flood fill and *Ch1* similarity; flood fill manages to eliminate many tiny clusters. Although the result is still not perfect, key components are correctly identified. (c) Chameleon 2 (with flood fill and improved similarity) but uses hMETIS partitioning, which ensures a clean result without noisy data points. However, the clusters on the right-hand side with different densities were not identified correctly. (d) Standard Chameleon 2 (flood fill and Ch2 similarity) with recursive bisection partitioning.

between all possible pairs of clusters, only the c nearest neighbors are considered as merging candidates. In the first iteration only $O(n \cdot c)$ are evaluated.

4.5 Improved Similarity Measure

The most significant modification we made is in the merging phase. During merging, Chameleon 1 chooses the most similar cluster pair and merges those paired clusters together; this step repeats until no pairs are left. The choice is based on a function that evaluates the proximity between every pair of clusters, the so called similarity measure.

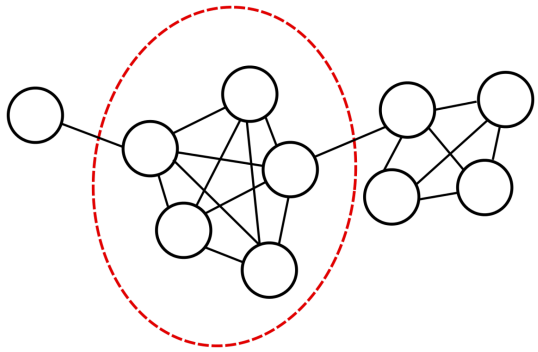


Fig. 6. An example of a balanced partitioning with a minimum amount of cut edges. The dashed line separates two clusters, which are the result of this partitioning. Both clusters contain the same number of nodes and only two edges were cut. However, part of the result is also a completely disconnected cluster that does not meet any definition of a good cluster and would cause problems in the merging phase.

Instead of relying on metrics computed from bisection we base similarity on the ratio between the average weight of all edges across two clusters and the sum of weights inside those clusters, denoted as

$$s(C_i) = \sum_{e \in C_i} w(e), \tag{6}$$

$$\bar{s}(C_i) = \frac{1}{|E_{C_i}|} s(C_i), \tag{7}$$

and $\bar{s}(C_i, C_j)$ is computed in the same manner using the average of the inter-cluster edge weights between cluster C_i and C_j. Thus, closeness is similar to the original one in Equation (3):

$$R_{CL2}(C_i, C_j) = \begin{cases} m_{\text{fact}} \cdot \dfrac{\bar{s}(C_i, C_j)}{s(C_i) + s(C_j)} & \text{for } |E_{C_i}| \vee |E_{C_j}| = 0 \\ \left(|E_{C_i}| + |E_{C_j}|\right) \cdot \dfrac{\bar{s}(C_i, C_j)}{s(C_i) + s(C_j)} & \text{for } |E_{C_i}| \wedge |E_{C_j}| > 0 \end{cases}. \tag{8}$$

Modified inter-connectivity considers only the ratio between the number of edges, instead of relying on edge weights (Equation (4)):

$$R_{\text{IC2}}(C_i, C_j) = \begin{cases} 1 & \text{for } |E_{C_i}| \vee |E_{C_j}| = 0 \\ \dfrac{|E_{C_{i,j}}|}{\min\left\{|E_{C_i}|, |E_{C_j}|\right\}} \cdot \rho(C_i, C_j)^{\beta} & \text{for } |E_{C_i}| \wedge |E_{C_j}| > 0 \end{cases}, \tag{9}$$

where $|E_{C_{i,j}}|$ represents the number of edges between clusters C_i and C_j. This part of the equation encourages merging of clusters connected by a large number of edges relative to the number of edges inside the clusters.

The last addition is the ρ factor, which discourages the algorithm from merging clusters with different densities: both clusters should have similar average between-point distances. The β parameter serves to modify the weight of the ρ factor:

$$\rho(C_i, C_j) = \frac{\min\{\bar{s}(C_i), \bar{s}(C_j)\}}{\max\{\bar{s}(C_i), \bar{s}(C_j)\}}. \tag{10}$$

For both inter-connectivity and closeness we have to handle a special case: A cluster containing an individual node. In that case, the cluster contains no edges, which leads to zero similarity. Such

clusters are often merged incorrectly, worsening the overall result. Even when the partitioning algorithm is configured to make strictly balanced partitions with the same number of items in each cluster, this problem persists. Therefore, this issue has to be handled during the merging phase.

When computing the similarity of a cluster pair where one of the clusters contains no edges, we compute only the pair's R_{CL2}. Then, we multiply the R_{CL2} by a constant m_{fact} to obtain the final cluster similarity. This process increases the similarity of all pairs containing single-item clusters and causes the clusters to merge with their neighbors in the early stages of the merging process, avoiding problems later on.

The resulting similarity is a product of modified closeness and inter-connectivity – as in the case of the original Chameleon. Then Chameleon 2 base similarity is defined as:

$$S_{\text{Ch2}}(C_i, C_j) = R_{\text{CL2}}(C_i, C_j)^{\alpha} \cdot R_{\text{IC2}}(C_i, C_j). \tag{11}$$

Similarity defined this way is faster to compute and in most cases provides better results than the original *Ch1* similarity (Equation (5)), as we show in Section 6.

When we incorporate all these improvements, the computational complexity of the algorithm is comparable to the original version. Below, we examine the complexity of these individual steps.

4.6 Algorithm Complexity

Chameleon 2 complexity is comparable to that of the previous version. Firstly a k-NN needs to be computed for the whole dataset, which has complexity of $\mathrm{O}(dn^2)$ with d being the complexity of computing distance between items (which is usually linear with respect to the number of dataset dimensions) and n being the number of data points. For each data point we need to visit all of its neighbors and compute the distance between the neighbor and the data point. In lower dimensions a kd-tree with $\mathrm{O}(d(\log(n)))$ complexity could be employed, however with growing dimensionality the kd-tree performance drops to a simple linear search.

The time needed to bisect all graphs at a single level is always $\mathrm{O}(n)$. The depth of the recursion depends on the desired number of partitions. In order to obtain m partitions, where $m = n/p_{\max}$, we need to recursively bisect the graph $\log(m)$ times. Thus, the recursive bisection complexity is $\mathrm{O}(n(\log(m)))$. The flood fill method requires a single scan of the graph, so the complexity is $\mathrm{O}(n)$.

Lastly, a naive merging process would have complexity $\mathrm{O}(m^3)$. By using a priority queue, we can reduce this complexity to $\mathrm{O}(m^2 \log(m))$ with m merging steps. Unlike *Ch1* similarity it is not necessary to bisect each merged cluster in order to compute new metric values. The *Ch2* similarity metric has linear update time.

Overall, the complexity of Chameleon 2 is $\mathrm{O}(dn^2 + n + (n + m^2)\log(m))$.

4.7 Parameter Settings

The algorithm parameters are listed with their default values in Table 1. In most cases, the default configuration provides a good enough solution. For setting k one should use $k = a + c\ln(n)$, where a and c are constants. These constants should be rather small integers because when k is too large, nodes might get connected with neighbors that are far away, leading to worse results.

5 CHAMELEON 2 AUTOPILOT

One problem with several state-of-the-art algorithms such as DBSCAN is their sensitivity to proper parameter setting. Also, resolving the number of clusters is a very complex task that many algorithms fail to perform well or do not address at all. In this section, we introduce a procedure called Autopilot that involves adaptive parameter setting and a smart cutoff mechanism to resolve the number of clusters autonomously.

Table 1. Default Parameters for Chameleon 2 Algorithm

Parameter	Description	Default value
k	Number of neighbors (k-NN)	$2\ln(n)$
α	Closeness priority	2.0
β	Inter-connectivity priority	1.0
$p_{\max}$	Max. partition size	$\max\{5, n/100\}$
similarity	Determines merging order	BBK
bisection	Bisection algorithm	Fiduccia–Mattheyses
m_{fact}	Factor for small clusters	10^3

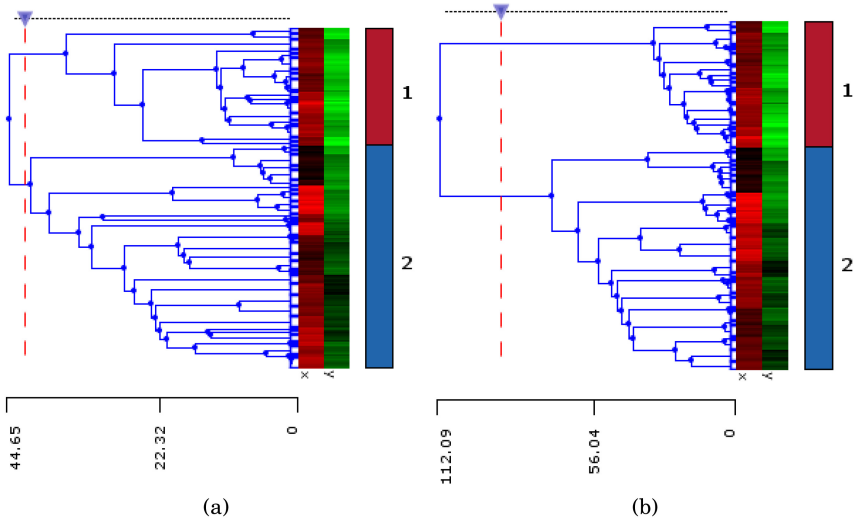


Fig. 7. Chameleon's dendrogram on the *flame* dataset using *Ch1* similarity (a) and *Ch2* similarity (b). It is easier to find a reasonable cutoff for dendrogram 7b automatically than for dendrogram 7a.

Chameleon 2 creates a hierarchical structure of merges. This result can be useful on its own as it enables us to explore several possible groupings. However, a typical use-case is that we want to obtain a single partitioning. To obtain that, we need to explore the hierarchical result and determine where the best single clustering lies.

5.1 Dendrogram Representation

A so called *dendrogram* is typically used to visualize the hierarchical structure, sometimes combined with a heatmap (an example is shown in Figure 7). In order to meaningfully display Chameleon's results, certain changes to the traditional dendrogram representation were made: instead of showing individual items, the lowest dendrogram level represents small clusters created in the partitioning phase. Chameleon's similarity works in the opposite way to a standard distance computation, therefore we use the inverse value of an edge's weight.

Finally, node height is redefined. Normally, the height of each node at ith level (each level represents a merge of two clusters) is defined as the distance between clusters C_x, C_y at levels x and y, which are then merged into the cluster C_i as

$$h(C_i) = d(C_x, C_y), \tag{12}$$

where $d(\cdot, \cdot)$ is a distance function between two clusters. We define the dendrogram's height recursively as

$$h(C_i) = \begin{cases} \frac{1}{\text{Sim}(C_x, C_y)}, & \text{for } i = 0 \\ h(C_{i-1}) + \frac{1}{\text{Sim}(C_x, C_y)}, & \text{for } i > 0 \end{cases}. \tag{13}$$

The main reason for this change is that over time, cluster similarity can increase, thus the distance decreases. With typical dendrogram representation this would cause the dendrogram to grow down, which would be confusing, and the search for a line that cuts the dendrogram would not make any sense. Furthermore, dendrograms using our definition are easier to read and automatic methods for partition extraction produce better results with this representation.

5.2 First Jump Cutoff

To find an optimal cut in the dendrogram structure we propose a simple, yet effective method named First Jump Cutoff (*Ch2-FJ*). The method is based on the idea that the first large gap between tree levels is the place where the clusters should be divided. This makes sense since the first big difference between tree levels signifies the merge of larger clusters that are distant, and thus not very similar. One of the clustering goals is to keep dissimilar clusters separated, therefore we try to cut the dendrogram as close to the tree root as possible.

To find the first jump, we compute the average distance between levels in the first half of the dendrogram. That means the average distance between clusters at the time of their merging during the first $n/2$ merges, where n is the total number of merges. After that, we look for the first distance between levels in the rest of the dendrogram, which is at least f times greater than the average we obtained from the first half. When found, the cut is made between these levels. If the condition if not fulfilled, we search for the first jump at least $\frac{f}{\eta^i}$ times bigger, where f, η are parameters and i is an iterator starting from 1, being increased by 1 until the aforementioned condition is fulfilled. The parameter f should be a large number, it defaults to 10^3; the decrease coefficient η is set to 2. Our experiments suggest that the algorithm is not sensitive to parameter changes. The whole method is illustrated in Algorithm 2.

A comparison between an ideal partitioning extraction method (*Ch2-**) and First Jump (*Ch2-FJ*) is shown in Table 3. On artificial datasets, the NMI score is decreased by 0.03 on average.

6 EXPERIMENTS

We evaluated Chameleon 2 against several popular clustering algorithms. We present the experiments in four subsections, the first of which explains our methodology. Section 6.2, then evaluates the modifications made to the Chameleon 2 algorithm. Section 6.3 compares the algorithm against other well-known clustering algorithms. We probe the capabilities and limitations of various clustering algorithms while optimizing the external score (based on supervised labels). Lastly, Section 6.4 follows the more realistic scenario of running clustering using only an unsupervised heuristic for parameter-guessing on high-dimensional data.

6.1 Methodology

The first part of our benchmark intentionally contains 2D and 3D datasets with clear human-distinguishable structures in the data; some datasets are contaminated with noise to test the robustness of the algorithm (see Table 2). All data files are available online.[5]

[5] https://github.com/deric/clustering-benchmark.

ALGORITHM 2: First Jump Cutoff

```
Function FirstJump(mult , factor)
    Data: Dendrogram tree
    Result: Tree height suggested for cutting
    avg ← ComputeAvgJump()
    res ← 0
    while mult > 0 do
        res ← FindBiggerJump(mult ·avg)
        if res ≠ 0 then
            return res
        else
            mult ←mult /factor
Function FindBiggerJump(jump)
    lower ← TreeHeightByLevel(treeLevels /2 + 1)
    for i ←treeLevels /2 to treeLevels do
        upper ← TreeHeightByLevel(i)
        if upper - lower > jump then
            return lower + (upper − lower )/2
        lower ← upper
    return 0
```

We included the same datasets that were used in the original Chameleon paper, however the datasets referred to as *DS1* and *DS2* were not available. The first one comes from CURE (Guha et al. 1998).[6] Instead of *DS2* the visually similar dataset *twodiamonds* was included. For CLUTO datasets (*t4.8k*, *t5.8k*, *t7.10k* – marked as *DS3* in Karypis et al. (1999b), *t8.8k* – *DS4* in Karypis et al. (1999b)), which did not contain any labels, we assigned labels based on our intuition.[7] Visualization of all datasets colored by reference labels can be found in the Supplementary material.

Another group of six datasets (*atom*, *chainlink*, *lsun*, *target*, *twodiamonds*, and *wingnut*) comes from the FCPS[8] (Ultsch and Mörchen 2005), which should serve as an elementary benchmark of clustering algorithms. The datasets were manually created in such a way that none of the traditional clustering algorithms would be able to solve them all.

On the *pathbased* dataset, which consists of an incomplete circle and two Gaussian distributed clusters inside, Chang and Yeung (2008) demonstrated the limitations of standard spectral clustering. Their proposed path-based spectral clustering method provides a better result, though still not perfect.

Evaluating the quality of clustering algorithms is hard, and numerous approaches have been applied in relevant studies. While unsupervised criteria can be used during clustering (Bartoň and Kordík 2015), their performance is data dependent. The prevailing approach is to evaluate clustering outcomes against external (ground truth) labels. In this article, we provide labels for every dataset in the benchmark (all datasets are listed in Table 2), therefore the quality of clustering can be evaluated using supervised criteria.

[6]Data was generated based on visual similarity to an image in the referred paper, we were unable to obtain the dataset from the original author.

[7]The labels were added manually, the original dataset did not contain any. A visualization of the assigned labels can be found in the Supplementary material.

[8]Fundamental Clustering Problems Suite.

Table 2. Datasets Used for Our Experiments: Most Contain Patterns Easily Distinguishable by Humans

Dataset	d	n	Noise size (%)	classes	Source
3-spiral	2	312	–	3	Chang and Yeung (2008)
aggregation	2	788	–	7	Gionis et al. (2007)
atom	3	800	–	2	Ultsch and Mörchen (2005)
chainlink	3	1,000	–	2	Ultsch and Mörchen (2005)
cluto-t4.8k	2	8,000	764 (9.55%)	7^7	Karypis (2002b)
cluto-t5.8k	2	8,000	1153 (14.41%)	9^7	Karypis (2002b)
cluto-t7.10k	2	10,000	792 (9.92%)	8^7	Karypis (2002b)
cluto-t8.8k	2	8,000	323 (4.04%)	8^7	Karypis (2002b)
compound	2	399	–	6	(Zahn 1971)
cure-t2-4k	2	4,000	200 (4.76%)	7	(Guha et al. 1998)[6]
D31	2	3,100	–	31	Veenman et al. (2002)
dense-disk-5k	2	5,000	–	2	Shatovska et al. (2007)[6]
DS-850	2	850	–	5	Su et al. (2005)
diamond9	2	3000	–	9	Salvador and Chan (2004)
disk-in-disk	2	4,600	–	2	Shatovska et al. (2007)[6]
dpb	2	4,000	657 (16.43%)	6^7	Rodriguez and Laio (2014)
flame	2	240	–	2	Fu and Medico (2007)
impossible	2	3,673	78 (2.12%)	8	Jain (2010)[6]
jain	2	373	–	2	Jain and Law (2005)
long1	2	1,000	–	2	Handl and Knowles (2007)
longsquare	2	900	–	6	Handl and Knowles (2007)
lsun	2	400	–	3	Ultsch and Mörchen (2005)
pathbased	2	300	–	3	Chang and Yeung (2008)
s-set1	2	5,000	–	15^7	Fränti and Virmajoki (2006)
sizes1	2	1,000	–	4	Handl and Knowles (2007)
smile1	2	1,000	–	4	Handl and Knowles (2007)
spiralsquare	2	1,500	–	6	Handl and Knowles (2007)
target	2	770	–	6	Ultsch and Mörchen (2005)
twodiamonds	2	800	–	2	Ultsch and Mörchen (2005)
wingnut	2	1,016	–	2	Ultsch and Mörchen (2005)
zelnik4	2	622	138 (22.19%)	5	Zelnik-Manor and Perona (2004)

Note: Eight datasets contain noisy clusters.

Normalized Mutual Information (NMI) comes from an Information Theory background (Kvålseth 1987) where it accounts for shared information. In a clustering context NMI computes agreement between cluster assignments and ground truth labels. Given two clusterings $\mathbb{C}$ and $\mathbb{C}'$, their entropies, joint entropy, conditional entropies and mutual information (MI) are defined via the marginal and joint distributions of data items in $\mathbb{C}$ and $\mathbb{C}'$, respectively, as Nguyen et al. (2010):

$$H(\mathbb{C}) = -\sum_{i=1}^{K} \frac{|C_i|}{n} \log \frac{|C_i|}{n}, \tag{14}$$

$$H(C, C') = -\sum_{i=1}^{K} \sum_{j=1}^{K'} \frac{M_{ij}}{n} \log \frac{M_{ij}}{n}, \tag{15}$$

$$H(\mathbb{C}|\mathbb{C}') = -\sum_{i=1}^{K}\sum_{j=1}^{K'}\frac{M_{ij}}{n}\log\frac{M_{ij}/n}{|C'_j|/n}, \tag{16}$$

$$I(\mathbb{C},\mathbb{C}') = \sum_{i=1}^{K}\sum_{j=1}^{K'}\frac{M_{ij}}{n}\log\frac{M_{ij}/n}{|C_i|\,|C'_j|/n^2}, \tag{17}$$

where M is the contingency table defined as $M_{ij} = C_i \cap C'_j$ and K, respective, K' is the number of clusters in $\mathbb{C}$, respective, $\mathbb{C}'$. Then, NMI_{sqrt} according to Strehl and Ghosh (2002) is defined as

$$f_{\text{NMI}_{\text{sqrt}}}(\mathbb{C},\mathbb{C}') = \frac{I(\mathbb{C},\mathbb{C}')}{\sqrt{H(\mathbb{C})\,H(\mathbb{C}')}}. \tag{18}$$

In the remainder of this article when we refer to NMI we always mean NMI_{sqrt}. An NMI value of 1.0 means complete agreement between clustering and external labels while 0.0 means the opposite. Another popular criterion for external evaluation is the Adjusted Rand Index (Hubert and Arabie 1985), which compares pairs of assignments from $\mathbb{C}$ and $\mathbb{C}'$, defined as

$$f_{\text{ARI}}(\mathbb{C},\mathbb{C}') = \frac{t_0 - t_3}{\frac{1}{2}(t_1+t_2) - t_3}$$

$$\text{where}\quad t_0 = \sum_{i=1}^{k}\sum_{j=1}^{l}\binom{M_{ij}}{2}$$

$$t_1 = \sum_{i=1}^{k}\binom{|C_i|}{2},$$

$$t_2 = \sum_{j=1}^{l}\binom{|C'_j|}{2},$$

$$t_3 = \frac{2t_1t_2}{n(n-1)}.$$

While each metric is based on a different concept, they produce very similar results, at least in our benchmark (see Figure 10 for ranking results).

It is not feasible to benchmark our algorithm against all existing algorithms. However, we have tried to select a representative algorithm from each of the main clustering method categories. For each algorithm and dataset, we have explored the configuration space exhaustively. The reported results should be very close to their NMI maximum. Some algorithms are very sensitive to initialization or parameter settings. In order to explore their limits, we iterated several different configurations for and chose the one that maximized the NMI.[9]

In order to reproduce the original Chameleon results, we used hMETIS binary,[10] which represents the most complex part of the algorithm. The remaining parts of the algorithm we implemented on our own, as described in Bruna (2015). We were unable to reproduce exactly the same results as were presented in the original study (Karypis et al. 1999b). However, after thorough tuning of the hMETIS parameters we managed to obtain a visually similar result (see Figure 8(a); the

[9]This approach is not possible when external labels are missing and provides a biased estimate of the performance. Real-life performance can be significantly lower without a clear optimization objective.

[10]For our experiments, we used the latest version hMETIS 2.0-pre from 2007.

(a) Ch1 – *cluto-t7.10k*, NMI = 0.88 (b) Ch2 – *cluto-t7.10k*, NMI = 0.89 (c) Ch2 – *cluto-t8.8k*, NMI = 0.94

Fig. 8. (a) Clustering produced by our Chameleon 1 implementation ($\alpha = 2$, $\beta = 1$, $k = 10$) on dataset *cluto-t7.10k* which uses hMETIS (`-ctype=gedge2 -rtype=fast -otype=cut -ufactor=6 -nruns=11`) and manual cutoff. There are several tiny clusters left after partitioning phase, total number of discovered clusters is 11. (b and c) Chameleon 2 result on datasets used in the original paper, the algorithm uses the same configuration in both cases, max. partition size = 10, $\alpha = 4$, $\beta = 1$ and manual cutoff.

Chameleon 2 result is shown in Figure 8(b)) with several noisy clusters that can be neglected due to the non-deterministic nature of the partitioning.

The k-means (Lloyd 1982) algorithm was provided with the "correct" number of ground truth classes (parameter k). Initialization is randomized and the NMI value is the maximum found in 100 independent runs.

HC is deterministic, so a single run is sufficient. From the hierarchical result we evaluated all possible clusterings and selected the solution with the maximal NMI value.

For DBSCAN (Ester et al. 1996), we followed the author's recommendation regarding ε estimation. First, we computed and sorted distances to the 4th nearest neighbor for each data point. Then using a simple heuristic we searched for an "elbow" in the distance values, which is typically located in the first third of distances sorted from highest to lowest value. From the elbow area we chose 10 different ε values and ran DBSCAN with $minPts = 4$; if no valid clustering was found, we increased $minPts$ by 1 and again tested different ε values until a meaningful clustering was found. From the clusterings produced we chose the one with the highest NMI value.

There is no heuristic known to us for configuration of the CURE algorithm. The most important parameter for influencing the quality of the clustering results is the shrink factor, which we set in the interval between 0.1 and 0.9 in 10 steps. For these experiments, we disabled random sampling, in order to obtain the highest possible clustering quality.

The CL-G algorithm was tested with 11 similarity functions, and we then chose the one with the highest NMI.

For Chameleon 2, we tested several configurations with different inter-connectivity and closeness priorities, variants with fixed k and fixed maximum partition size were also included (10 variants in total). From the resulting dendrogram we evaluated all possible cuts and chose the one that maximized NMI.

6.2 Evaluation of Improvements

Several improvements were introduced in Chameleon 2. Table 3 shows algorithm performance when certain modifications are turned off. In the first column ($Ch2^*$), we observe Chameleon 2 performance with all improvements enabled. The remaining columns display the difference between the NMI obtained by this default (fully improved) Chameleon 2 configuration and the best NMI found using a version of Chameleon 2 without the given modification. A positive number signifies that the given configuration was able to produce better clustering than the default Chameleon 2. For each Chameleon 2 run, we evaluated several promising configurations.

Table 3. Evaluation of Some Ch2 Improvements

Dataset	Ch2*	Ch2-asym	Ch2-hMETIS	Ch2-std	Ch2-noff	Ch2-FJ
3-spiral	1.00	0.00	0.00	0.00	**−0.05**	0.00
aggregation	0.99	0.00	**−0.02**	**−0.01**	**−0.03**	**−0.10**
atom	1.00	0.00	0.00	0.00	0.00	0.00
chainlink	1.00	0.00	0.00	0.00	0.00	0.00
cluto-t4.8k	0.89	**−0.01**	**−0.01**	0.00	**−0.02**	**−0.01**
cluto-t5.8k	0.86	**−0.03**	**−0.04**	0.00	**−0.06**	**−0.03**
cluto-t7.10k	0.91	**−0.05**	**−0.04**	0.00	**−0.05**	**−0.07**
cluto-t8.8k	0.94	**−0.01**	0.00	+0.01	**−0.07**	0.00
compound	0.99	**−0.04**	**−0.08**	0.00	**−0.01**	**−0.04**
cure-t2-4k	0.97	**−0.06**	**−0.07**	+0.01	**−0.04**	**−0.02**
D31	0.96	0.00	0.00	**−0.02**	**−0.04**	**−0.01**
dense-disk-5k	0.91	**−0.06**	**−0.13**	**−0.41**	**−0.03**	**−0.02**
diamond9	0.99	0.00	+0.01	**−0.01**	**−0.02**	0.00
disk-in-disk	0.99	**−0.07**	**−0.20**	**−0.24**	**−0.03**	**−0.40**
dpb	0.81	**−0.01**	**−0.03**	**−0.05**	+0.01	**−0.07**
DS-850	0.98	+0.02	+0.01	**−0.01**	**−0.02**	0.00
flame	0.93	+0.01	+0.03	**−0.02**	**−0.10**	0.00
impossible	0.97	**−0.01**	**−0.01**	**−0.01**	**−0.06**	**−0.05**
jain	1.00	0.00	0.00	0.00	**−0.03**	0.00
long1	1.00	0.00	0.00	0.00	0.00	0.00
longsquare	0.98	+0.01	**−0.01**	0.00	**−0.01**	0.00
lsun	1.00	0.00	0.00	0.00	0.00	0.00
pathbased	0.89	**−0.02**	+0.03	**−0.01**	**−0.05**	**−0.03**
s-set1	1.00	0.00	**−0.02**	**−0.01**	**−0.04**	0.00
sizes1	0.91	+0.02	**−0.01**	**−0.01**	**−0.05**	0.00
smile1	1.00	0.00	0.00	0.00	0.00	0.00
spiralsquare	0.99	0.00	**−0.04**	**−0.01**	**−0.04**	**−0.05**
target	1.00	**−0.06**	**−0.06**	0.00	**−0.05**	0.00
triangle1	1.00	0.00	0.00	0.00	0.00	0.00
twodiamonds	1.00	0.00	0.00	0.00	0.00	0.00
wingnut	1.00	0.00	0.00	0.00	0.00	0.00
zelnik4	0.99	**−0.07**	**−0.05**	0.00	**−0.02**	0.00
AVG	0.96	−0.01	−0.02	−0.03	−0.03	−0.03

Note: The maximum NMI value found for each Ch2 without the given improvement. For each clustering solution, all possible cutoffs are evaluated (except *Ch2-FJ*). *Ch2** best Chameleon 2 clustering found with the highest possible supervised criteria *Ch2-asym* uses asymmetrical *k*-NN. *Ch2-hMETIS* uses hMETIS partitioning, with more degrees of freedom than recursive bisection. We evaluated 80 configurations of Ch2 with hMETIS. In bold type are highlighted results where the method gives worse results than the proposed Chameleon 2 algorithm. *Ch2-std* computes similarity in the same manner as Chameleon 1. *Ch2-noff* does not include the flood fill step (reconnecting isolated points). *Ch2-FJ* extracts final partitioning from the dendrogram in an unsupervised manner using *FirstJump* method.

The idea was to use different k estimators, giving preference to either inter-connectivity or closeness as those parameters are hard to estimate without prior data knowledge.

hMETIS provides better results than recursion bisection especially on datasets with low-density points. However, the overall results favor recursive bisection. The most significant difference is observed on *disk-in-disk* and *dense-disk-5k* datasets – data with varying density. It is apparent that

recursive bisection improved the performance of Chameleon on complex datasets (*zelnik4*, *target*, *disk-in-disk*, etc.), where it is hard to find a crisp boundary between clusters.

As far as the quality of results is considered, Fiduccia-Mattheyses and hMETIS provide comparable results. A naive recursive bisection method does not perform consistently with growing data size, thus more robust approaches like hMETIS should be preferred.

In clustering quality terms the most significant improvements are introduced by flood fill (*Ch2-noff* show results without flood fill) and the redefined similarity measure (*Ch2-std* uses original Chameleon 1 similarity).

Ch2-asym with asymmetrical *k*-NN performs better on datasets with fewer data points. On larger datasets the symmetrical approach leading to smaller number or edges brings better results.

Ch2-FJ runs with all improvements enabled, however for cluster extraction from the dendrogram the automated First Jump heuristic is used, running in fully unsupervised mode. The results suggest that for most of the datasests First Jump managed to find the solution with the highest NMI. The worst performance decrease was observed for the *disk-in-disk* datasets where clusters are not clearly separated from each other. The average performance decrease in NMI was 0.03.

6.3 Chameleon 2 Benchmark

Finally, we performed a comprehensive benchmark on a wide range of clustering problems used in the literature to compare Chameleon 2 to other clustering algorithms. Table 4 gives the NMI values for 10 clustering methods and 32 datasets; the best results obtained are highlighted in bold type. Chameleon 2 ended up achieving the highest average NMI score over all datasets; in most cases the algorithm is capable of providing clustering results closest to the optimal solution. There are a few exceptions when other algorithms are capable of discovering better clustering. *Wingnut* dataset consists of two rectangle areas with gradient data point distribution. Chameleon 2 assigns bordering points incorrectly, partially due to the small number of data points in the area. A similar issue occurs in the case of the *disk-in-disk* dataset, where points from clusters with lower density are incorrectly assigned to clusters with higher density (see Figure 9(b)). Table 5 shows the same benchmark measured by the Adjusted Rand Index.

The no free lunch theorem also applies to data clustering – there is no algorithm that would be perfect on all datasets. Surprisingly good results were produced by DBSCAN, for which parameters are difficult to guess. When optimizing the external objective, one can almost always find a configuration that is nearly optimal for given dataset. Despite the good results it should be noted that DBSCAN requires a configuration heuristic and usually multiple runs are needed to obtain a valid clustering. In many cases, all points are assigned to a single cluster, however it is hard to predict invalid configuration without actually running the algorithm.

CL-G provides fast and accurate clustering results on most of the tested datasets. The algorithm requires that the expected number of clusters be defined minus connected components in the graph, which might be difficult to estimate for unknown data. CL-G clusters the *disk-in-disk* dataset with the lowest error rate of all the tested algorithms. However, it fails on the *dense-disk-5k* dataset, which was generated in a very similar manner, but without space between the inner disk and outer ring (see Figure 9(a)).

Although the CW algorithm was initialized the same way as Chameleon, the results are quite different. This outcome is not very surprising because CW is designed for processing huge networks. Obviously using graph representation without appropriate similarity metric definition is not sufficient for pattern recognition.

Multi-objective Chameleon (*Ch-MO*) works well on many datasets, however the inability of setting preference to either connectivity or closeness limits the algorithm's performance on the more complex problems included in the benchmark. The main strength of the multi-objective approach

Table 4. The NMI Values of Clusterings Produced with the Best Configuration Found for Each Dataset

Dataset	Ch2*	DBSCAN	Ch-MO	HC-SL	CL-G	Ch1	CURE	*k*-means	CW	AP
3-spiral	**1.000**	**1.000**	**1.000**	**1.000**	0.132	0.697	0.220	0.001	0.687	0.016
aggregation	**0.992**	0.948	**0.992**	0.895	0.961	0.753	0.982	0.845	0.895	0.741
atom	**1.000**	0.994	**1.000**	**1.000**	0.911	0.649	0.414	0.304	0.438	0.028
chainlink	**1.000**	**1.000**	**1.000**	**1.000**	0.879	**1.000**	0.631	0.069	0.417	0.340
cluto-t4.8k	0.893	**0.909**	0.873	0.861	0.877	0.851	0.618	0.582	0.578	0.460
cluto-t5.8k	0.864	**0.934**	0.824	0.805	0.836	0.764	0.822	0.799	0.574	0.369
cluto-t7.10k	0.912	**0.927**	0.832	0.871	0.857	0.752	0.624	0.574	0.604	0.531
cluto-t8.8k	**0.944**	0.881	0.890	0.864	0.918	0.808	0.631	0.550	0.598	0.494
compound	**0.992**	0.923	0.917	0.848	0.872	0.731	0.860	0.679	0.873	0.753
cure-t2-4k	**0.967**	0.876	0.882	0.820	0.834	0.841	0.755	0.666	0.539	0.554
D31	0.957	0.849	**0.958**	0.869	0.957	0.929	0.944	0.922	0.754	0.627
dense-disk-5k	**0.908**	0.660	0.497	0.458	0.298	0.453	0.281	0.000	0.303	0.002
diamond9	0.993	0.956	0.992	0.986	0.997	0.871	0.983	**1.000**	0.691	0.572
disk-in-disk	**0.990**	0.698	0.244	0.663	0.941	0.203	0.216	0.000	0.241	0.003
dpb	0.810	**0.844**	0.725	0.754	0.667	0.722	0.746	0.682	0.382	0.526
DS-850	0.984	0.971	0.995	0.975	0.987	0.818	**1.000**	0.890	0.748	0.644
flame	**0.927**	0.816	0.912	0.836	0.899	0.711	0.794	0.454	0.518	0.465
impossible	**0.969**	0.932	0.936	0.939	0.874	0.899	0.803	0.702	0.600	0.796
jain	**1.000**	0.832	**1.000**	0.856	**1.000**	0.732	**1.000**	0.357	0.433	0.235
long1	**1.000**	0.989	**1.000**	**1.000**	**1.000**	0.972	**1.000**	0.002	0.413	0.000
longsquare	0.981	0.944	**0.993**	0.942	0.986	0.861	0.810	0.866	0.750	0.551
lsun	**1.000**	**1.000**	**1.000**	**1.000**	0.987	0.980	**1.000**	0.534	0.645	0.682
pathbased	**0.887**	0.725	0.847	0.704	0.552	0.716	0.515	0.549	0.555	0.356
s-set1	**0.997**	0.958	0.979	0.968	0.988	0.981	0.972	0.942	0.870	0.756
sizes1	0.909	0.774	0.912	0.813	0.730	0.802	0.066	**0.926**	0.620	0.769
smile1	**1.000**	**1.000**	**1.000**	**1.000**	0.882	0.884	0.772	0.608	0.672	0.631
spiralsquare	**0.987**	0.965	0.986	0.922	0.926	0.641	0.598	0.561	0.644	0.491
target	**1.000**	0.986	0.940	**1.000**	0.657	0.785	**1.000**	0.703	0.524	0.017
triangle1	**1.000**	**1.000**	**1.000**	**1.000**	0.900	0.938	**1.000**	0.952	0.704	0.742
twodiamonds	**1.000**	0.987	**1.000**	0.934	**1.000**	0.810	0.902	**1.000**	0.414	0.975
wingnut	**1.000**	**1.000**	0.990	**1.000**	**1.000**	0.901	0.174	0.770	0.441	0.700
zelnik4	0.987	**0.994**	0.890	0.786	0.899	0.676	0.620	0.742	0.606	0.712
AVG	**0.964**	0.915	0.906	0.886	0.850	0.785	0.711	0.601	0.585	0.486
σ (SD)	0.049	0.095	0.160	0.123	0.200	0.160	0.280	0.312	0.161	0.277

Note: These values represent the upper bounds of the clusterings that can be discovered by each algorithm. The best result(s) on the given dataset are highlighted in bold type. The difference between the means of Ch2 and DBSCAN are statistically very significant – the two-tailed p-value of the paired t-test is equal to 0.0015.

is the possibility of using a custom data-specific objective function. Nonetheless in these experiments we used the same combination of objectives for all datasets.

Table 4 summarizes the experiments on the artificial datasets benchmark. The difference between the score means of Chameleon 2 and DBSCAN, which has the second best average score, is statistically very significant – the two-tailed p-value of the paired t-test is equal to 0.0015. The difference between Chameleon 2 and Multi-objective Chameleon, the third best on average, is also statistically significant with the p-value of 0.0338. The remaining differences between the means of Chameleon 2 and other algorithms are at least as significant as the difference from DBSCAN.

Table 5. Results on the Very Same Benchmark as in Table 4 Only Measured by Adjusted Rand Index

Dataset	Ch2*	DBSCAN	HC-SL	Ch-MO	CL-G	Ch1	CURE	*k*-means	AP	CW
3-spiral	**1.000**	**1.000**	**1.000**	**1.000**	0.020	0.598	0.032	0.005	0.003	0.404
aggregation	**0.995**	0.950	0.809	**0.995**	0.958	0.671	0.988	0.711	0.629	0.809
atom	**1.000**	0.998	**1.000**	**1.000**	0.924	0.574	0.347	0.193	0.008	0.072
chainlink	**1.000**	**1.000**	**1.000**	**1.000**	0.878	**1.000**	0.656	0.093	0.243	0.042
cluto-t4.8k	0.922	**0.951**	0.940	0.862	0.877	0.832	0.466	0.440	0.328	0.188
cluto-t5.8k	0.859	**0.945**	0.885	0.777	0.791	0.733	0.779	0.746	0.198	0.186
cluto-t7.10k	0.947	0.947	**0.950**	0.778	0.813	0.645	0.433	0.338	0.366	0.195
cluto-t8.8k	**0.969**	0.888	0.758	0.870	0.926	0.728	0.504	0.328	0.359	0.235
compound	**0.997**	0.955	0.944	0.907	0.858	0.679	0.831	0.515	0.712	0.854
cure-t2-4k	**0.986**	0.811	0.806	0.880	0.732	0.905	0.655	0.418	0.399	0.176
D31	0.933	0.632	0.634	**0.937**	0.933	0.866	0.875	0.818	0.193	0.411
dense-disk-5k	**0.954**	0.860	0.767	0.411	0.279	0.579	0.319	0.000	0.001	0.007
diamond9	0.995	0.956	0.985	0.993	0.998	0.814	0.976	**1.000**	0.356	0.394
disk-in-disk	**0.997**	0.945	0.950	0.012	0.976	0.003	0.267	0.000	0.000	0.002
dpb	0.855	**0.889**	0.888	0.760	0.632	0.767	0.784	0.696	0.486	0.249
DS-850	0.987	0.979	0.980	0.997	0.990	0.764	**1.000**	0.893	0.529	0.542
flame	**0.967**	0.891	0.912	0.950	0.950	0.784	0.854	0.465	0.437	0.284
impossible	**0.976**	0.929	0.967	0.940	0.815	0.847	0.642	0.611	0.634	0.284
jain	**1.000**	0.923	0.938	**1.000**	**1.000**	0.731	**1.000**	0.300	0.124	0.070
long1	**1.000**	0.996	**1.000**	**1.000**	**1.000**	0.984	**1.000**	0.001	−0.001	0.045
longsquare	0.984	0.954	0.948	**0.995**	0.989	0.800	0.637	0.798	0.265	0.558
lsun	**1.000**	**1.000**	**1.000**	**1.000**	0.995	0.990	**1.000**	0.422	0.694	0.330
pathbased	**0.907**	0.696	0.592	0.876	0.468	0.728	0.424	0.464	0.209	0.390
s-set1	**0.997**	0.955	0.968	0.966	0.982	0.973	0.925	0.834	0.467	0.759
sizes1	0.938	0.805	0.869	0.939	0.595	0.811	0.004	**0.956**	0.718	0.305
smile1	**1.000**	**1.000**	**1.000**	**1.000**	0.813	0.877	0.700	0.547	0.602	0.323
spiralsquare	**0.996**	0.990	0.958	**0.996**	0.863	0.320	0.404	0.320	0.321	0.266
target	**1.000**	**1.000**	**1.000**	0.970	0.626	0.856	**1.000**	0.657	0.004	0.225
triangle1	**1.000**	**1.000**	**1.000**	**1.000**	0.855	0.938	**1.000**	0.963	0.674	0.385
twodiamonds	**1.000**	0.995	0.968	**1.000**	**1.000**	0.859	0.941	**1.000**	0.990	0.049
wingnut	**1.000**	**1.000**	**1.000**	0.996	**1.000**	0.944	0.052	0.856	0.563	0.085
zelnik4	0.990	**0.996**	0.827	0.889	0.867	0.635	0.436	0.679	0.644	0.461
AVG	**0.973**	0.932	0.914	0.897	0.825	0.757	0.654	0.533	0.380	0.299
σ (SD)	0.040	0.089	0.107	0.200	0.225	0.202	0.313	0.316	0.264	0.225

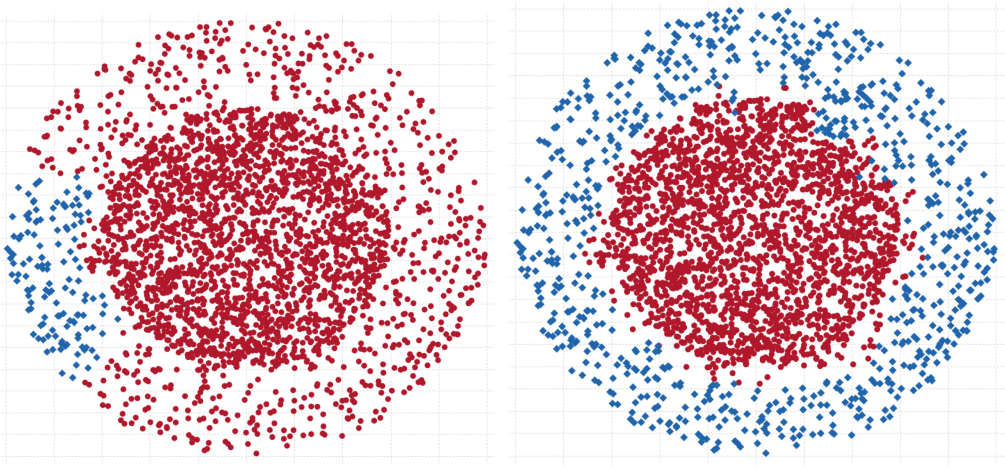
Note: The difference between the means of Ch2 and DBSCAN is statistically very significant – the two-tailed p-value of the paired t-test is equal to 0.0015.
Bold face are highlighted best Adjusted Rand scores found on given dataset in our benchmark.

Finally, we measured the performance of Chameleon 2 on high-dimensional real world datasets that are very different from the artificial data used above.

6.4 Results on High-Dimensional Data

The Olivetti Faces[11] is a collection of 400 photos in 64 × 64 resolution, having 40 different people each photographed 10 times with varying perspective or face expression. Each image is

[11] http://www.cs.nyu.edu/~roweis/data.html.



(a) CLUTO, NMI = 0.18 (b) Ch2, NMI = 0.76

Fig. 9. (a) CLUTO fails to discover clusters on dataset *dense-disk* with different densities that are not clearly separated. (b) Chameleon 2 with higher preference of inter-connectivity ($\alpha = 2, \beta = 4$) identifies both clusters correctly.

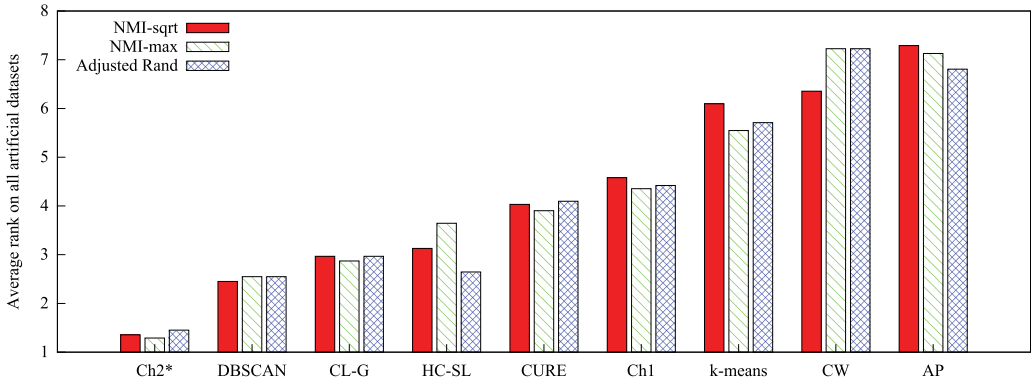


Fig. 10. Average clustering methods ranking over all datasets in the benchmark for several supervised metrics (lower is better). The external metric does influence overall performance, however the first two positions remain unchanged.

represented as a 4096 vector with grayscale pixels having values between 0 and 255. Chameleon 2 found 44 clusters with NMI = 0.77 (see Figure 11).

Table 6 shows results on real-world problems. We tested Chameleon 2 on single-cell mass cytometry data of bone marrow mononuclear cells (Bendall et al. 2011) and obtained high NMI values between 0.8 and 0.9.

The results suggest that the proposed algorithm is applicable to a wide range of clustering problems.

7 CONCLUSION

We introduced an advanced clustering algorithm Chameleon 2 that improves several drawbacks found in the previous version from Karypis et al. (1999b). The results of our proposed approach

Fig. 11. A subsample of clusters found in the Olivetti faces dataset that contains images of 400 faces (photos of 40 different people). Chameleon 2 found 44 clusters – NMI = 0.77. Each cluster is colored by a different color, faces on input are in grayscale.

Table 6. Benchmark on Real-World Problems

Dataset	n	d	Ch2*	Ch2-FJ	Time (s)
pen digits (Lichman 2013)	10,992	16	0.87	0.87	20.88
cytof.h2 (Bendall et al. 2011)	31,721	32	0.98	0.98	194.98
cytof.h1 (Bendall et al. 2011)	72,463	32	0.98	0.85	1019.35
cytof.one (Bendall et al. 2011)	81,747	13	0.88	0.81	751.21
MNIST (Lecun and Cortes 2013)	70,000	784	0.82	0.76	5289.48

Note: Results were evaluated using NMI_{sqrt}. Running time was computed as an average of 10 independent Ch2-FJ runs, which included time needed for computing k-NN as well.

agree well with human judgment. To confirm the robustness of the algorithm, a comprehensive benchmark was executed over a suite consisting of 32 datasets representing various segmentation problems as well as on high-dimensional real-world datasets.

Chameleon 2 brings a novel flood fill step after the partitioning phase, an alternative partitioning method and a redefined similarity measure. Together these enhancements ensure better results and higher clustering stability. A method for automatic cluster extraction from the Chameleon 2 dendrogram is also proposed.

Our experiments show that replacing (closed-source) hMETIS partitioning is possible without any significant drop in clustering quality. On most datasets recursive bisection provides better results that come at the cost of higher computation time.

Unlike ensemble approaches, meta-algorithms or evolutionary algorithms, Chameleon 2 belongs with traditional methods that produce single, high-quality results comparable to clusterings produced by far more complex algorithms with high computational complexity.

Chameleon 2 is a fully automated algorithm that produces high-quality clustering on diverse datasets without the need to configure dataset-specific parameters. The default algorithm settings work on a wide range of data with a minimal error rate. Moreover, by configuring each phase of the algorithm, Chameleon 2 is able to adapt its cluster definitions for practically any dataset. Therefore, Chameleon 2 can also be viewed as a general clustering framework that is applicable to a wide range of complex clustering problems and performs well on datasets commonly used in the clustering literature.

7.1 Future Work

Finding exactly k nearest neighbors is the most expensive part of the algorithm. Locality-sensitive hashing (LSH), split tree, or similar methods could be used to find approximate nearest neighbors. These methods trade clustering quality for running time. Finding a reasonable compromise and approximation configuration for given data would be interesting, especially for larger datasets.

There is still room for improvement during the partitioning phase where faster algorithms could be employed without significantly affecting the final clustering result. Also, Chameleon 2 does not handle noise.

Chameleon 2 was designed for datasets where humans can find reasonable-looking clusters in spite of variable data density, complex distributions, and the presence of noise. The biggest advantage of our approach is that it can solve various datasets efficiently without expert interaction and parameter settings. It is very likely that efficient clustering for high dimensional or mixed variable datasets will involve other clustering approaches. We would like to explore the field of ensembling and meta-learning in the data clustering domain to develop a more universal clustering framework.

ACKNOWLEDGMENT

We would like to thank Petr Bartůněk, Ph.D. from the IMG CAS for supporting our research and permitting us to publish the details of our work.

REFERENCES

Charu C. Aggarwal and Chandan K. Reddy (Eds.). 2014. *Data Clustering: Algorithms and Applications*. CRC Press.

G. Ball and D. Hall. 1965. *ISODATA: A Novel Method of Data Analysis and Pattern Classification*. Technical Report. Stanford Research Institute, Menlo Park.

T. Barton, T. Bruna, and P. Kordik. 2016. MoCham: Robust hierarchical clustering based on multi-objective optimization. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW'16)*. 831–838.

Tomáš Bartoň and Pavel Kordík. 2015. Evaluation of relative indexes for multi-objective clustering. In *Hybrid Artificial Intelligent Systems*, Enrique Onieva, Igor Santos, Eneko Osaba, Héctor Quintián, and Emilio Corchado (Eds.), Lecture Notes in Computer Science, vol. 9121. Springer International Publishing, 465–476.

Sean C. Bendall, Erin F. Simonds, Peng Qiu, D. Amir El-ad, Peter O. Krutzik, Rachel Finck, Robert V. Bruggner, Rachel Melamed, Angelica Trejo, Olga I. Ornatsky, Robert S. Balderas, Sylvia K. Plevritis, Karen Sachs, Dana Pe'er, Scott D. Tanner, and Garry P. Nolan. 2011. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. *Science* 332, 6030 (2011), 687–696.

Chris Biemann. 2006. Chinese whispers – An efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: The 2nd Workshop on Graph Based Methods for Natural Language Processing*. New York City, 73–80.

M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich. 1997. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters* 35, 1 (15 Aug. 1997), 33–42.

Tomas Bruna. 2015. *Implementation of the Chameleon Clustering Algorithm*. Bachelor's Thesis. CTU in Prague.

Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3, 1 (1974), 1–27.

Rich Caruana, Mohamed Elhawary, Nam Nguyen, and Casey Smith. 2006. Meta clustering. In *Proceedings of the 6th International Conference on Data Mining (ICDM'06)*. IEEE Computer Society, Washington, DC, 107–118.

Hong Chang and Dit-Yan Yeung. 2008. Robust path-based spectral clustering. *Pattern Recognition* 41, 1 (2008), 191–203.

David L. Davies and Donald W. Bouldin. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1979), 224–227.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B (Methodological)* 39, 1 (1977), 1–38.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad (Eds.). AAAI Press, 226–231.

B. S. Everitt. 1993. *Cluster Analysis*. Edward Arnold.

C. M. Fiduccia and R. M. Mattheyses. 1982. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference (DAC'82)*. IEEE Press, Piscataway, NJ, 175–181.

Pasi Fränti and Olli Virmajoki. 2006. Iterative shrinking method for clustering problems.*Pattern Recognition* 39, 5 (2006), 761–775.

Brendan J. J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315, 5814 (Jan. 2007), 972–976. DOI: https://doi.org/10.1126/science.1136800

Limin Fu and Enzo Medico. 2007. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics* 8 (2007), 3.

Junhao Gan and Yufei Tao. 2015. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 519–530.

Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. 2007. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data* 1, 1 Article 4 (March 2007), 30 pages. http://doi.acm.org/10.1145/1217299.1217303

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Record*, Vol. 27. ACM, 73–84.

J. Handl and J. Knowles. 2007. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation* 11, 1 (2007), 56–76.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification* 2, 1 (1985), 193–218.

Anil K. Jain. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters* 31, 8 (2010), 651–666.

Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ.

Anil K. Jain and Martin H. C. Law. 2005. Data clustering: A user's dilemma. In *Pattern Recognition and Machine Intelligence*, Sankar K. Pal, Sanghamitra Bandyopadhyay, and Sambhunath Biswas (Eds.), Lecture Notes in Computer Science, vol. 3776. Springer, 1–10.

Anil K. Jain, Alexander Topchy, Martin H. C. Law, and Joachim M. Buhmann. 2004. Landscape of clustering algorithms. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*. IEEE Computer Society, Washington, DC, 260–263.

Raymond A. Jarvis and Edward A. Patrick. 1973. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers* 100, 11 (1973), 1025–1034.

George Karypis. 2002a. *CLUTO A Clustering Toolkit*. Technical Report 02-017. Department of Computer Science, University of Minnesota.

George Karypis. 2002b. Karypis Lab - CLUTO's Datasets. Retrieved from http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.

George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1999a. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration Systems* 7, 1 (Mar. 1999), 69–79.

George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. 1999b. Chameleon: Hierarchical clustering using dynamic modeling. *Computer* 32, 8 (Aug. 1999), 68–75.

George Karypis and Vipin Kumar. 1999. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference (DAC'99)*. 343–348.

L. Kaufman and P. J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. A John Wiley & Sons, Inc.

Tarald O. Kvålseth. 1987. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics* 17, 3 (1987), 517–519.

G. N. Lance and W. T. Williams. 1967. A general theory of classificatory sorting strategies. *Computer Journal* 9, 4 (1967), 373–380. arXiv:http://comjnl.oxfordjournals.org/content/9/4/373.full.pdf+html.

Yann Lecun and Corinna Cortes. 2013. The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist.

M. Lichman. 2013. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.

Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedngs of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Le Cam and J. Neyman (Eds.), vol. 1. University of California Press, 281–297.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software* 2, 11 (2017), 205.

G. J. McLachlan and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.

Mark E. J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (2004), 026113.

Xuan Vinh Nguyen, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11 (2010), 2837–2854.

Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science* 344, 6191 (2014), 1492–1496.

Stan Salvador and Philip Chan. 2004. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proceedings of the16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*. IEEE Computer Society, 576–584.

Tetyana Shatovska, Tetiana Safonova, and Iurii Tarasov. 2007. A modified multilevel approach to the dynamic hierarchical clustering for complex types of shapes. In *Proceedings of the ISTA (LNI)*, Heinrich C. Mayr and Dimitris Karagiannis (Eds.), vol. 107. GI, 176–186.

Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles – A knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research* 3 (Dec. 2002), 583–617.

Mu-Chun Su, Chien-Hsing Chou, and Chen-Chiung Hsieh. 2005. Fuzzy C-means algorithm with a point symmetry distance. *International Journal of Fuzzy Systems* 7, 4 (2005), 175–181.

Robert Tibshirani, Walther Guenther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society Series B* 63, 2 (2001), 411–423.

Alfred Ultsch and Fabian Mörchen. 2005. *ESOM-Maps: Tools for Clustering, Visualization, and Classification with Emergent SOM*. Technical Report No. 46. Department of Mathematics and Computer Science, University of Marburg, Germany.

Cor J. Veenman, Marcel J. T. Reinders, and Eric Backer. 2002. A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 9 (2002), 1273–1280.

C. T. Zahn. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers* C-20, 1 (Jan. 1971), 68–86.

Lihi Zelnik-Manor and Pietro Perona. 2004. Self-tuning spectral clustering. In *Proceedings of Neural Information Processing Systems (NIPS'04)*. 1601–1608.

Y. Zhao and G. Karypis. 2001. *Criterion Functions for Document Clustering: Experiments and Analysis*. Technical Report TR 01–40. Department of Computer Science, University of Minnesota, Minneapolis, MN.

Received April 2017; revised July 2018; accepted November 2018