computation

MDPI

Article

Clustering Improves the Goemans–Williamson Approximation for the Max-Cut Problem

Angel E. Rodriguez-Fernandez [1,2,*], Bernardo Gonzalez-Torres [3], Ricardo Menchaca-Mendez [4], and Peter F. Stadler [1,2,5,6,7,*]

1 Max Planck Institute for Mathematics in the Sciences, 04103 Leipzig, Germany
2 Bioinformatics Group, Department of Computer Science, Universität Leipzig, 04107 Leipzig, Germany
3 Department of Computer Science, University of California, Santa Cruz, CA 95064, USA; beaugonz@ucsc.edu
4 Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City 07738, Mexico; ric@cic.ipn.mx
5 Institute for Theoretical Chemistry, University of Vienna, 1090 Wien, Austria
6 Facultad de Ciencias, Universidad National de Colombia, Sede Bogotá 111321, Colombia
7 Santa Fe Insitute, Santa Fe, NM 87501, USA
* Correspondence: angel@bioinf.uni-leipzig.de (A.E.R.-F.); studla@bioinf.uni-leipzig.de (P.F.S.)

Received: 29 June 2020; Accepted: 24 August 2020; Published: 26 August 2020

Abstract: MAX-CUT is one of the well-studied NP-hard combinatorial optimization problems. It can be formulated as an Integer Quadratic Programming problem and admits a simple relaxation obtained by replacing the integer "spin" variables x_i by unitary vectors $\vec{v}_i$. The Goemans–Williamson rounding algorithm assigns the solution vectors of the relaxed quadratic program to a corresponding integer spin depending on the sign of the scalar product $\vec{v}_i \cdot \vec{r}$ with a random vector $\vec{r}$. Here, we investigate whether better graph cuts can be obtained by instead using a more sophisticated clustering algorithm. We answer this question affirmatively. Different initializations of k-means and k-medoids clustering produce better cuts for the graph instances of the most well known benchmark for MAX-CUT. In particular, we found a strong correlation of cluster quality and cut weights during the evolution of the clustering algorithms. Finally, since in general the maximal cut weight of a graph is not known beforehand, we derived instance-specific lower bounds for the approximation ratio, which give information of how close a solution is to the global optima for a particular instance. For the graphs in our benchmark, the instance specific lower bounds significantly exceed the Goemans–Williamson guarantee.

Keywords: algorithms; approximation; semidefinite programming; Max-Cut; clustering

1. Introduction

The MAX-CUT problem is a well-known NP-hard [1] and APX-hard [2] combinatorial optimization problem. An instance consists of a graph G with vertex set V, edge set E and edge weights $w : E \to \mathbb{R}$. A cut is a bipartition $A, V \setminus A$ of the vertex set V, where $A \subseteq V$. The MAX-CUT problem consists of finding a cut that maximizes the total weight of the edges that span the cut. That is, the function

$$f(A) = \sum_{\{p,q\} \in E : p \in A, q \in V \setminus A} w(\{p,q\}) \tag{1}$$

is to be maximized over all $A \subseteq V$. While MAX-CUT is NP-hard in general, polynomial-time algorithms exist for restricted graph classes, including planar graphs [3], graph without long odd cycles [4], and cographs [5]. MAX-CUT can also be written as a spin glass model, see, e.g., [6], for an overview. Using an arbitrary numbering of the n vertices of the graph G, we write $x_i = +1$ if vertex $i \in A$,

$x_i = -1$ if vertex $i \in V \setminus A$, and $x = (x_1, x_2, \dots, x_n)$. Furthermore, we set $w_{ij} := w(\{i,j\})$ if $\{i,j\}$ is an edge of G and $w_{ij} = 0$ otherwise. With this notation, we can rewrite $f(A)$ in terms of the "spin vector" as

$$f(x) = \frac{1}{4}\sum_{i,j} w_{ij}(1 - x_i x_j) \qquad x_i \in \{-1, +1\} \tag{2}$$

Maximizing $f(x)$ amounts to the integer quadratic programming (IQP) formulation of the MAX-CUT problem. Without loss of generality, we assume $w_{ij} = w_{ji}$.

Solutions of large MAX-CUT problems are of considerable practical interest in network design, statistical physics, and data clustering, and hence a broad array of computational techniques have been customized to its solution. Broadly, they can be subdivided into combinatorial heuristics (see e.g., [7–9] and the references therein) and methods involving relaxations of the integer constraints in Equation (2).

Replacing the integer vectors by $x \in \mathbb{R}^n \setminus \{0\}$ leads to an equivalent continuous optimization problem for which an excellent heuristic is described in [10] and a close connection with the largest eigenvalues of generalized Laplacians [11] and their corresponding eigenvectors [12]. On this basis, algorithms with uniform approximation guarantees have been devised [13,14]. Goemans and Williamson [15] replaced the integers x_i by unitary vectors $\vec{v}_i$ of dimension $n = |V|$.

$$\begin{aligned} &\max_{\vec{v}_i} \frac{1}{4}\sum_{i,j} w_{ij}\left(1 - \vec{v}_i \cdot \vec{v}_j\right) \\ &\text{subject to } |\vec{v}_i| = 1 \quad \forall i = 1, \dots, n \end{aligned} \tag{3}$$

where $\cdot$ denotes the scalar product on the unitary vectors. This problem contains all the instances of the original problem (2), as seen by setting $\vec{v}_i = (x_i, 0, \dots, 0)$ for all i. The relaxed problem (3) is a particular instance of Vector Programming, or using a change of variables, an instance of Semidefinite Programming (SDP), and thus can be solved in polynomial time (up to an arbitrarily small additive error), see, e.g., [16,17].

The solutions of the relaxed problem (3) are translated to solutions of the original IQP. Goemans and Williamson [15] proposed to use a unitary random vector $\vec{r}$ and to set

$$\hat{x}_i = \operatorname{sgn}(\vec{v}_i \cdot \vec{r}) \tag{4}$$

with $\operatorname{sgn}(t) = -1$ for $t < 0$ and $\operatorname{sgn}(t) = +1$ for $t \geq 0$. This amounts to cutting the sphere at the hyperplane with normal vector $\vec{r}$ and to assign x_i depending on whether $\vec{v}_i$ lies in the "upper" or "lower" hemisphere defined by this hyperplane.

The Goemans–Williamson relaxation yields an approximation bound of $\alpha := \min_{0 \leq \theta \leq \pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} > 0.878$ for the expected value $\mathbb{E}[f(\hat{x})]/\max f$, where the expectation is taken over the choices of $\vec{r}$ [15]. At present, it is the best randomized approximation algorithm for the integer quadratic programming formulation of the MAX-CUT problem. A clever derandomization [18] shows that deterministic algorithms can obtain the same approximation bound. On the other hand, it is NP-hard to approximate MAX-CUT better than the ratio 16/17 [19]. If the Unique Games Conjecture [20] is true, furthermore, the approximation ratio cannot be improved beyond the Goemans–Williamson bound for all graphs. However, better ratios are achievable, e.g., for (sub)cubic graphs [21].

The translation of the solution of the related problem (3) in the Goemans–Williamson approximation relies on the choice of a random vector $\vec{r}$. Naturally, we ask whether the performance can be improved by expending more efforts to obtain a better choice for $\vec{r}$. The key observation is that the purpose of $\vec{r}$ is to separate the solution vectors $\vec{v}_i$ of Equation (3) into two disjoint sets of points on the sphere. The two sets $A_+ := \{i : \vec{v}_i \cdot \vec{r} \geq 0\}$ and $A_- := \{i : \vec{v}_i \cdot \vec{r} < 0\}$ can thus be thought of as a pair of clusters. Indeed, two vectors $\vec{v}_i$ and $\vec{v}_j$ tend to be anti-parallel if w_{ij} is large, while pairs of points i and j with small or even negative weights w_{ij} are likely to wind up on the same side of the maximal cut. Of course, the random vector $\vec{r}$ is just one way of expressing this idea: if $\vec{v}_i$ and $\vec{v}_j$ are similar, then we will "usually" have $\operatorname{sgn} \vec{v}_i \cdot \vec{r} = \operatorname{sgn} \vec{v}_j \cdot \vec{r}$. The "randomized rounding" of the Goemans–Williamson method therefore can also be regarded as a clustering method. This immediately begs the questions whether the solutions of the MAX-CUT problem can be improved by replacing the random choice of $\vec{r}$ by first clustering the solution set $\{\vec{v}_i\}$ of the relaxed problem (3). We shall see *empirically* that the answer to this question is affirmative even though the theoretical performance bound is not improved.

In practical applications, solutions of relaxed problems are often post-processed by local search heuristics. Therefore, a local search starting from the final results of both the Goemans–Williamson relaxation and two of our best clustering approaches were made in order to improve the cut values.

This contribution is organized as follows. In the following section, we briefly summarize data sets and clustering algorithms with their relevant properties as well as the details of the local search used to improve the cut values. In Section 3.1, we describe an initial analysis of the effect of clustering on the cut weights, showing that the quality of near-optimal clusters correlates well with cut weights. Since we were not able to show for most clustering methods that they retain the Goemans–Williamson performance bound, we derive an instance specific bound in Section 3.2 that provides a convenient intrinsic quality measure. In Section 3.3, we extend the empirical analysis to the benchmarking set that also contains very large graphs. We show that the use of clustering methods indeed provides a consistent performance gain. We also see that the instance-specific performance bounds are much closer to 1 than the uniform Goemans–Williamson α. Finally, in Section 3.4, we consider the improvement to the cut values that are achieved with local search starting from the the Goemans–Williamson and the two best clustering relaxations.

2. Materials and Methods

2.1. Benchmark Data

In order to assess the utility of clustering as rounding method, we used the benchmark set of graphs generated using Rinaldi's machine-independent graph generator. Both the generator and the graphs can be downloaded from Ye's web page http://web.stanford.edu/~yyye/yyye/Gset/. These graphs vary from 800 to 20,000 nodes and have edge weights of ± 1. The topology of the graph can be toroidal, almost planar or random. The first 21 G-set graphs are a standard benchmark for the MAX-CUT problem. The G-set benchmark consists of graphs G1 to G67, G70, G72, G77, and G81. The optimal cuts are not known for most of these graphs. We therefore use the best known cut-values compiled in [9] for comparison.

The relaxed problem (3) was solved using the `CVX` package in Matlab for graphs G1 to G21. For graphs G22 to G54, G57 to G59, G62 to G67, and G72, we used Mathematica's function `SemidefiniteOptimization`. For graphs G55, G56, G60, G61, G70, G77, and G81 neither `SemidefiniteOptimization` nor `CVX` were able to find a solution to the SDP problem. We therefore had to exclude these instances from further consideration. From the SDP solution of each instance, we computed 50 iterations for the randomized clustering algorithms, including Goeamans and Williamson randomized clustering and reported the best solution for the seven best algorithms.

2.2. Clustering in the Goemans–Williamson Algorithm

We consider the following clustering methods:

- **randomized rounding** as defined in [15].
- ***k*-means** [22] adapted to the unitary sphere as in [23] for fixed $k = 2$.
- ***k*-medoids** [24] for fixed $k = 2$.
- **Fuzzy *c*-means** [25] adapted to the unitary sphere as in [23] for fixed $c = 2$.
- **Minimum Spanning Tree** (MST) clustering, i.e., splitting the MST at the longest edge [26].

In order to obtain the spherical variants of k-means and Fuzzy c-means, it is necessary to define the cluster centroids as points on the sphere. This can be achieved by rescaling the centroid to be unit length [23]. On the sphere, cosine similarity is a more natural choice than Euclidean dissimilarity. It is not difficult to see that the two variants are actually equivalent:

Lemma 1. *Spherical k-means clustering with cosine similarity is equivalent to k-means clustering with Euclidean distances and normalizing of the centroid vectors in each step.*

Proof. For unitary vectors, the square of their Euclidean distance can be expressed as

$$\|\vec{x}-\vec{y}\|^2 = \vec{x}\cdot\vec{x} + \vec{y}\cdot\vec{y} - 2\vec{x}\cdot\vec{y} = 2(1-\cos\theta)$$

in terms of the angle θ between $\vec{x}$ and $\vec{y}$. The centroid $\vec{c}$ of a given cluster $\mathcal{C}$ minimizes $\sum_{i\in\mathcal{C}} \|\vec{x}_i - \vec{c}\|^2$ and thus maximizes the sum $\sum_{i\in\mathcal{C}} \cos\theta_i$. Analogously, each x_i is assigned to cluster $\mathcal{C}_j$ with minimal value of $\|\vec{x}_i - \vec{c}_j\|^2$ and thus maximal $\cos\theta_i$. Thus, the squared Euclidean distance and the cosine distance optimize the same objective function. □

The same argument can be made for Fuzzy c-means clustering, since its cost function is also a linear combination of squared Euclidean distances and thus, equivalently, a linear combination of the corresponding cosines. For MST clustering, no adjustment is necessary since the relationship between Euclidean distances and cosines is monotonic and thus the transformation does not affect the MST. By the same token, k-medoids clustering is unaffected by the restriction to the sphere since medoids by definition are always the unitary vectors $\vec{v}_i$.

An important ingredient for the clustering procedures is the initialization. For k-means, k-medoids, and fuzzy c-means, we consider both a deterministic and a non-deterministic version. In the deterministic version, the pair $\vec{v}_i^*$ and $\vec{v}_j^*$ with maximal distance from each other is chosen. In the non-deterministic variant, the two initial cluster centroids are selected from the solution vectors $\vec{v}_i$ at random. For k-means, we observed that choosing the initial cluster centroids as vectors $\vec{v}_i$ does not work well since the optimization quickly get stuck in a local minimum. We therefore devised two alternative random initialization methods: (1) We generate a random unitary vector $\vec{r}$ and use $\vec{c}_1 = \vec{r}$ and $\vec{c}_2 = -\vec{r}$ as initial centroids. (2) We use two independently generated random unitary vectors $\vec{r}_1$ and $\vec{r}_2$ as initial centroids. The main advantage of method (1) is that this initialization is equivalent to starting k-means from solutions obtained by Goemans–Williamson rounding. To see this, assume (without loss of generality) that the random vector chosen as initial centroid points towards the north pole and therefore the negative side of it will point towards the south pole. Then, any point in the lower hemisphere of the hypersphere will be clustered with the south pole vector since this is the closest centroid. The same will happen for points on the upper hemisphere and therefore this is equivalent to splitting the hypershpere into two hemispheres, which is the Goemans–Williamson rounding.

In summary, we consider a total of nine clustering procedures:

1. Fuzzy c-means **(Fuzzy)**
2. Randomized k-means initialized among vectors $\vec{v}_i$ **(K-MeansRand)**
3. Deterministic k-means **(K-MeansDet)**
4. Randomized k-medoids **(K-MedRand)**
5. Deterministic k-medoids **(K-MedDet)**
6. Minimum Spanning Tree **(MST)**
7. Randomized Rounding of Goemans–Williamson **(RR)**
8. Randomized k-means initialized with two random vectors **(K-Means2N)**
9. Randomized k-means initialized with a random vector and its negative **(K-MeansNM)**

In order to quantify the quality of the clusters, we use the *distortion* [27,28], a measure of cluster coherence defined as

$$\mathrm{dis}(\mathcal{C}) = \sum_{j=1}^{k} \sum_{\vec{x}_i \in C_j} ||\vec{x}_i - \vec{\mu}_{C_j}|| \tag{5}$$

Here, $\vec{\mu}_{C_j} := \frac{1}{|C_j|}\sum_{\vec{x}_i \in C_j} \vec{x}_i$ is the centroid of the cluster $C_j \in \mathcal{C}$. We note that k-means clustering minimizes the distortion.

2.3. Local Search

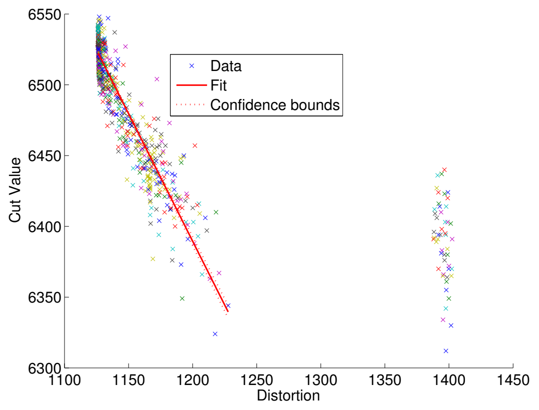
A natural definition of locality for cuts considers two cuts $A \neq A'$ *adjacent* if $A = A' \cup \{u\}$ for some $u \in V \setminus A'$ or $A' = A \cup \{u\}$ for some $u \in V \setminus A$ [12]. In terms of the spin vectors x, this amounts to "flipping" (changing the sign of) exactly one spin x_i. An adaptive walk iteratively accepts a spin flip if the cut value $f(A)$ improves. By construction, therefore, an adaptive walk terminates in a locally optimal solution, i.e., in a cut A^* for which there is no adjacent cut with a strictly larger cut weight. In general, neither the Goemans–Williamson nor any of the clustering results are locally optimal. We therefore use adaptive walks as a simple way to further improve solutions. We performed local improvement for each of the 50 repetitions of Goemans–Williamson randomized rounding, and the two best-performing clustering algorithms: K-MeansNM and K-Means2N.

3. Results

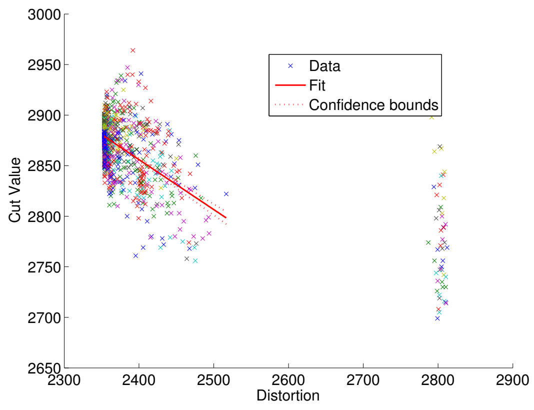
3.1. Cluster Quality Correlations with Solution Quality

In a preliminary evaluation on the first 21 G-set graphs (see Materials and Methods), we observed that clustering instead of random rounding yields systematically larger cuts. In order to better understand the reason for the beneficial effect of clustering, we investigated the relationship between a quality measure for the clustering and the resulting weight of the maximal cut. Since k-means clustering minimizes distortion, it serves as a natural measure of cluster quality, irrespective of the clustering method that is actually used. We chose K-MeansNM for this initial analysis because it uses RR solutions to initialize clusters and thus allows for a direct evaluation of the effect of clustering heuristics.

The RR solutions fall into a very narrow range of distortion values that is clearly separated from the near optimal range achievable by the clustering methods. The cut weights of the RR solutions do not appear to be correlated with the distortion of the corresponding clusters. However, after only a few clustering steps, k-means enters a regime in which distortion and cut weight are strongly correlated, see Figure 1.



(**a**) Scatter plot of 50 runs of K-MeansNM for graph G43.



(**b**) Scatter plot of 50 runs of K-MeansNM for graph G31.

Figure 1. Exploration of the path of 50 iterations of K-MeansNM on the distortion weight of the corresponding cut. The diagram shows all values generated while running the k-means. Points at the bottom right of the plot are the starting points and thus the results of Goemans–Williamson rounding; meanwhile, points at the top left are the end points. At each step of K-MeansNM, the points move to the left until the algorithm finds a local minima of distortion. The red line is a linear fit of all the points after the second step of k-means. These show a clear correlation between cluster distortion and cut weight.

Figure 1 provides a clear motivation to consider clustering as means of improving the Goemans–Williamson solution. We observe that there are two groups of graphs. In the first groups, exemplified by G43, Figure 1a, we consistently observe lower RR values at the starting point of k-MeansNM (right) that at the endpoint (top left) and the cut values mostly increase monotonically with decreasing distortion. In the second group of graphs, exemplified by G31, Figure 1b, this is still the case *on average*; however, the optimal cut weights are observed at sub-optimal distortions. This observation motivates us to record the cut weights for intermediary steps of the cluster procedures, not only at their endpoints. In the case of K-MeansNM, this guarantees that we retain the performance guarantee of the Goemans–Williamson bound.

3.2. An Instance-Specific Approximation Guarantee

Considering a fixed instance of MAX-CUT, let $\{\vec{v}_i\}$ be the solution of the relaxed problem (3) and let $\{A, V \setminus A\}$ be a discrete solution. Denote by $\partial A := \{(i,j) \in E(G) | i \in A, j \in V \setminus A\}$ the set of cut edges. The value S of the relaxed solution can be written as

$$\begin{aligned} S &= \frac{1}{2} \sum_{(i,j)\in\partial A} w_{ij} \left(1 - \vec{v}_i \cdot \vec{v}_j\right) + \frac{1}{2} \sum_{(i,j)\notin\partial A} w_{ij} \left(1 - \vec{v}_i \cdot \vec{v}_j\right) \\ &= \underbrace{\sum_{(i,j)\in\partial A} w_{ij}}_{f(A)} - \underbrace{\frac{1}{2} \sum_{(i,j)\in\partial A} w_{ij} \left(1 + \vec{v}_i \cdot \vec{v}_j\right)}_{g_{cut}} + \underbrace{\frac{1}{2} \sum_{(i,j)\notin\partial A} w_{ij} \left(1 - \vec{v}_i \cdot \vec{v}_j\right)}_{g_{in}} . \end{aligned}$$

Thus, we have $f(A) = S + g_{cut} - g_{in}$. Writing f^* for the weight of the optimal cut, we know that the solution of the relaxed problem is an upper bound, i.e., $S \geq f^*$. We therefore have

$$f(A) \geq f^* - (g_{in} - g_{cut}) \tag{6}$$

Note that $g_{in} - g_{cut} \geq 0$ since by definition $f^* \geq f(A)$. First, consider the case of positive edge-weights. Then, $f(A) > 0$ and we can estimate the approximation ratio for the solution A as

$$\alpha(A) := \frac{f(A)}{f^*} \geq 1 - \frac{g_{in} - g_{cut}}{f^*} \geq 1 - \frac{g_{in} - g_{cut}}{f(A)} \tag{7}$$

If there are negative edge weights, we follow [15], define $W_- := \sum_{(i,j)} \min(w_{ij}, 0) \leq 0$, and make use of the fact that $f^* - W_- \geq 0$. From Equation (6), we immediately obtain $f(A) - W_- \geq f^* - W_- - (g_{in} - g_{cut})$ and thus a generalized version of the approximation ration can be computed as

$$\alpha(A) := \frac{f(A) - W_-}{f^* - W_-} \geq 1 - \frac{g_{in} - g_{cut}}{f^* - W_-} \geq 1 - \frac{g_{in} - g_{cut}}{f(A) - W_-} \tag{8}$$

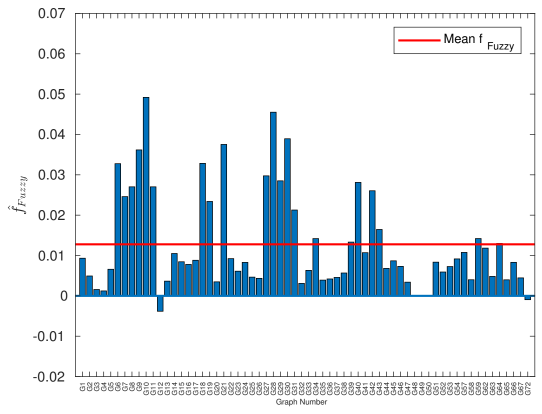
The bounds in Equations (7) and (8) can be seen as instant-specific versions of the general approximation ratio derived in [15]. Empirically, we found in benchmarking experiments (see below) that the instance specific bound $\alpha(A)$ substantially exceeds the uniform Goemans–Williamson bound of $\alpha \approx 0.878$. For the Goemans–Williamson algorithms, of course, we necessarily have $\mathbb{E}[\alpha(A)] \geq \alpha$.

3.3. Benchmarking Experiments

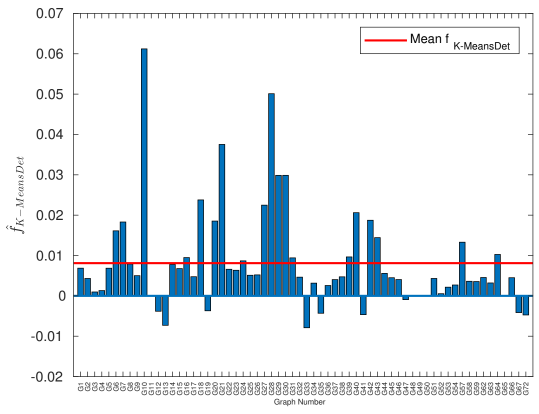
In order to compare the cut values of RR with each of the clustering algorithms, we use the following relative measure of performance:

$$\hat{f}_{cluster} = \frac{f_{cluster} - f_{RR}}{f_{RR}} = \frac{f_{cluster}}{f_{RR}} - 1 \tag{9}$$

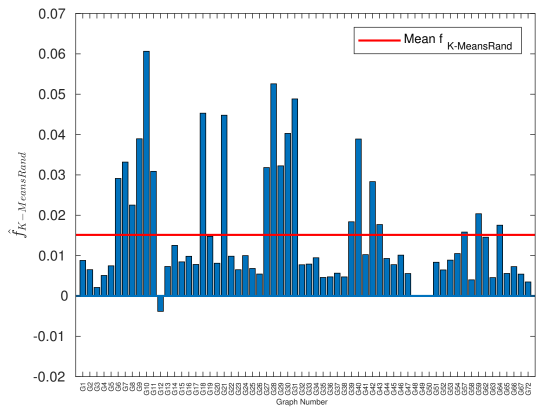
Figure 2 shows that clustering on most instances yield significant improvement for most clustering approaches. K-MeansNM by construction always finds a solution that is at least as good as RR; however, both K-MeansNM and K-Means2N *never* give a lower solution than RR. K-meansRand improves for all graphs except G12, Fuzzy for all except G12 and G72, K-MedRand for all except G32, G39, and G72, and, finally, for K-MeansDet in nine graphs, a better solution was not found.



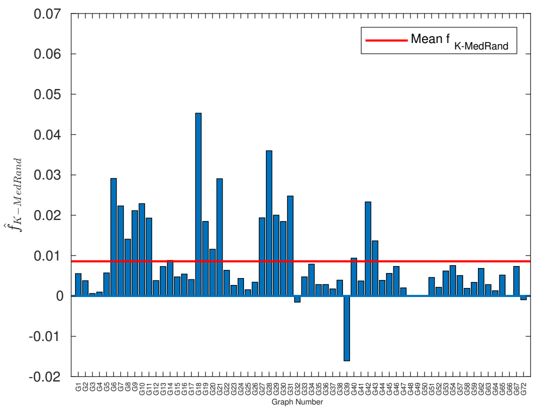
(**a**) Plot of $\hat{f}_{cluster}$ for Fuzzy clustering.



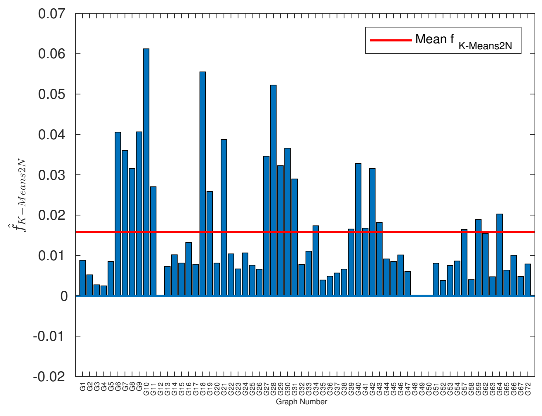
(**b**) Plot of $\hat{f}_{cluster}$ for K-MeansDet clustering.



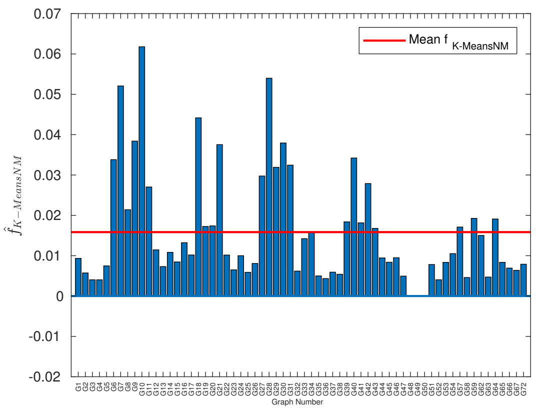
(**c**) Plot of $\hat{f}_{cluster}$ for K-MeansRand clustering.



(**d**) Plot of $\hat{f}_{cluster}$ for K-MedRand clustering.



(**e**) Plot of $\hat{f}_{cluster}$ for K-Means2N clustering.

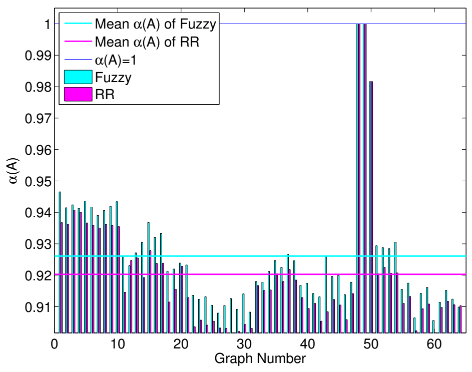


(**f**) Plot of $\hat{f}_{cluster}$ for K-MeansNM clustering.

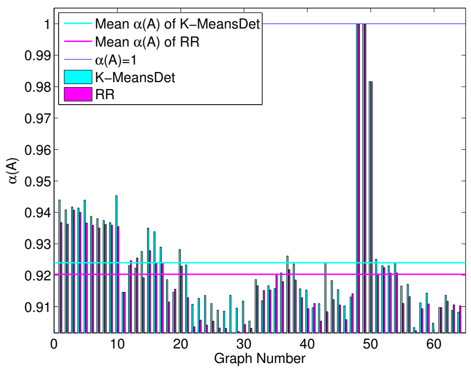
Figure 2. Comparison between the cut value found by clustering algorithms and RR, using $\hat{f}_{cluster}$ as the comparison. The horizontal axis represents the graph number, i.e., graph G*i* is shown in position *i*. The red line indicates the average over the benchmark set. Positive values indicate that the clustering solutions are superior to the RR solutions.

The gains in solution quality differ substantially between the test instances, and a few graphs, in particular G12 and G72, do not profit from the clustering approaches other than the randomized versions of K-means. Interestingly, these two graphs are toroidal.

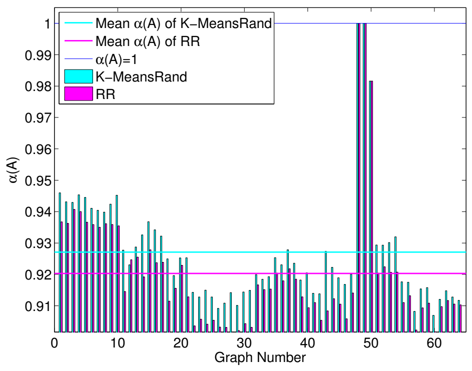
The same trend is also observed for the instance-specific performance bounds, see Figure 3. The performance bounds for the individual solutions are well above 0.9, i.e., exceed the Goemans and Williamson also in those few cases where clustering is worse than RR. For bipartite graphs with non-negative edge weights, the entire edge set forms a maximal cut [11]. This is the case for the two unweighted graphs G48 and G49 and explains why, for these two cases, no difference between RR and clustering solutions is observed in Figure 2.



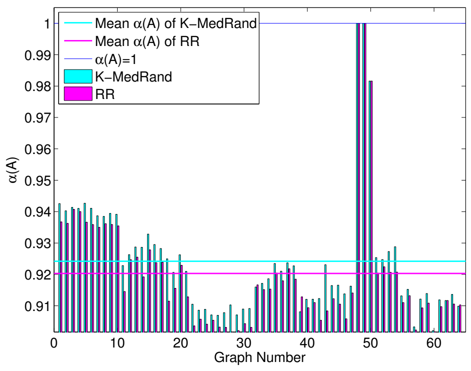
(**a**) Comparison of $\alpha(A)$ for Fuzzy clustering and RR.



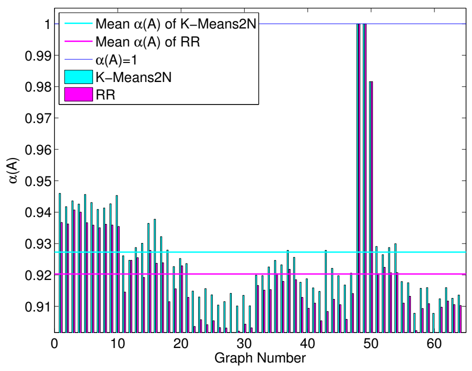
(**b**) Comparison of $\alpha(A)$ for K-MeansDet and RR.



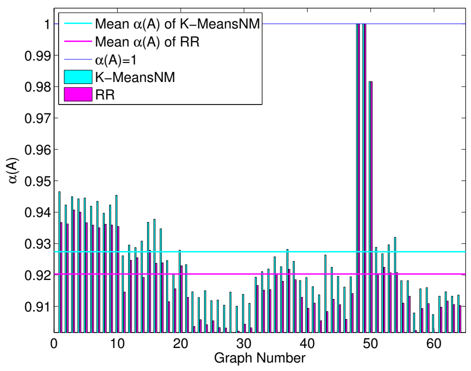
(**c**) Comparison of $\alpha(A)$ for K-MeansRand and RR.



(**d**) Comparison of $\alpha(A)$ for K-MedRand and RR.



(**e**) Comparison of $\alpha(A)$ for K-Means2N and RR.



(**f**) Comparison of $\alpha(A)$ for K-MeansNM and RR.

Figure 3. Comparison of instance-specific performance bounds $\alpha(A)$ between clustering algorithms and RR. The cyan line is the average of $\alpha(A)$ for the clustering methods and the magenta line is the average of RR for comparison.

On average, all clustering algorithms yield improvements over RR. Interestingly, the average solution quality depends noticeably on the clustering algorithm. The clustering algorithm with largest average improvement was K-MeansNM, as expected. On the benchmark set, an average increase on the cut weight by $\hat{f}_{\text{K-MeansNM}} = 1.541\%$ compared with RR is obtained. Other variants of 2-means performed similarly well. We found $\hat{f}_{\text{K-Means2N}} = 1.533\%$ and $\hat{f}_{\text{K-MeansRand}} = 1.471\%$. For Fuzzy clustering, we only observed an improvement of $\hat{f}_{\text{Fuzzy}} = 1.247\%$. With $\hat{f}_{\text{K-MedRand}} = 0.841\%$ and $\hat{f}_{\text{K-MeansDet}} = 0.789\%$, performance of medoids and deterministic methods was less encouraging. For individual instances, the improvement was substantial. For example, we obtain an improvement of 5.81% for the graph G10 and 5.12% for G28 with K-MeansNM.

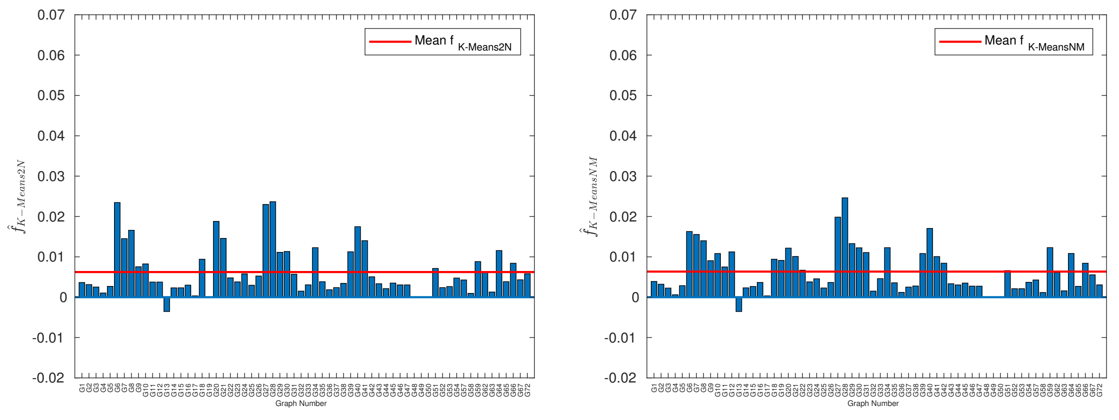
Despite subtle differences between the clustering methods, the clusters are quite similar in their characteristics. One measure of interest is the *mean clustering angle*

$$\theta_A := \arccos\left(\left|\frac{1}{|A|}\sum_{v_i \in A} v_i\right|\right) \tag{10}$$

It measures the average angle between two unit vectors in the same cluster. We found that the cardinalities $|A|$ and $|X \setminus A|$ are nearly even and θ_A lies between 60 and 75 degrees for the graphs in the benchmark set. We observed not convincing trends connecting these parameters and the weight of maximum cut.

3.4. Local Search Improvement

A straightforward way to improve a given solution of MAX-CUT is to add a local search step. We use adaptive walks for this purpose and restrict ourselves to RR and the two best-performing clustering approaches, i.e., K-MeansNM and K-Means2N. The results presented in Figure 4 are the best solutions found in all of the 50 iterations for each algorithm. The same relative measure of performance, $\hat{f}_{cluster}$, is used as in Section 3.3. The corresponding instance-specific performance bound α can be found in the Additional Material.



(**a**) Plot of $\hat{f}_{cluster}$ for K-Means2N clustering after local search.

(**b**) Plot of $\hat{f}_{cluster}$ for K-MeansNM clustering after local search.

Figure 4. Comparison between the cut value found by clustering algorithms and RR after local search (LS), using $\hat{f}_{cluster}$ as the comparison. The horizontal axis represents the graph number, i.e., graph G*i* is shown at position *i*. The red line indicates the average over the benchmark set. Positive values indicate that the locally improved clustering solutions are superior to the locally improved RR solutions.

Figure 5 shows the ratio $f_{rounding}/f_{best}$, where $f_{rounding}$ is the best solution after local search found either by clustering or RR for that instance and f_{best} is the best solution known in the literature, taken from [9]. The corresponding values are also available in tabular form as Additional Material. Graphs G11 to G13, G32 to G34, G57, G62, G65 to G67, and G72 have toroidal topology; both the clustering algorithm and Randomized Rounding show a comparably low approximation ratio for these instances. For graphs with other topologies, we obtain approximation ratios exceeding 0.96, sometimes closely approaching 1. The combination of the Goemans–Williamson relaxation with clustering thus at least comes close to the best solutions known in the literature.

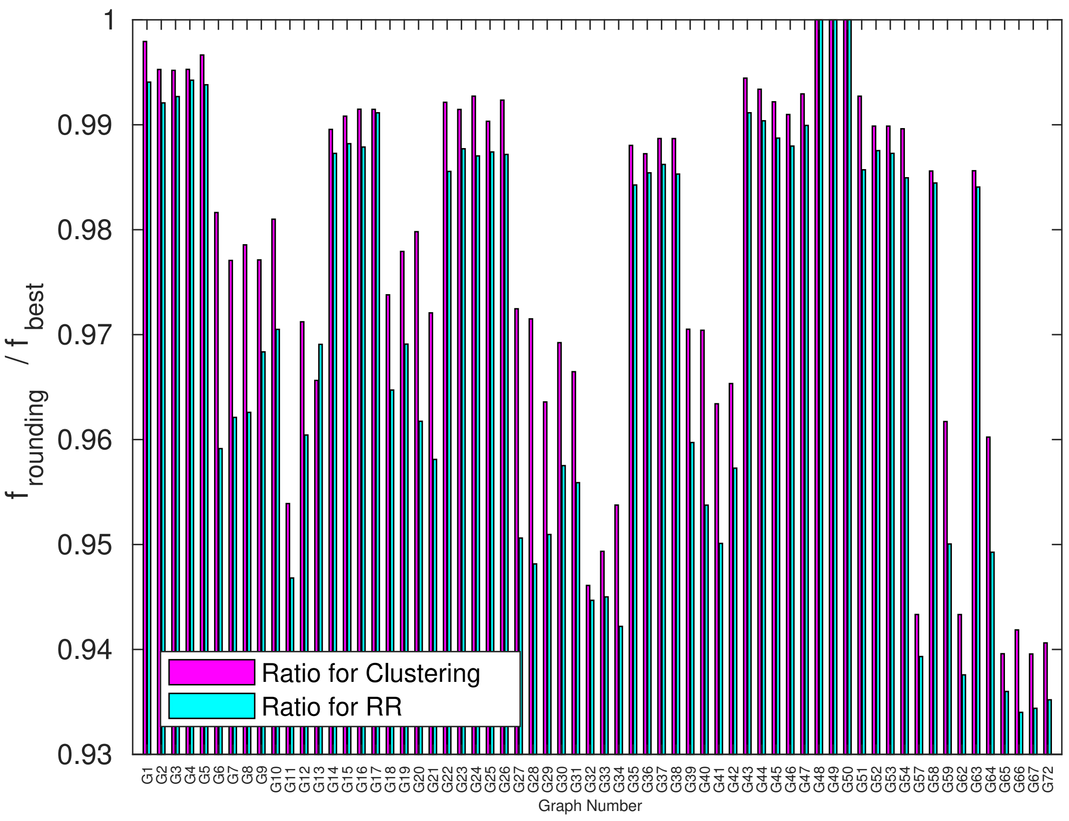


Figure 5. Comparison of the best cut value obtained with clustering (in magenta) and RR (in cyan) and subsequence local improvement with the best cut value (f_{best}) available in the literature.

4. Conclusions

As we can see from the results, using other clustering methods than the randomized version of [15], on average, leads to better cut values. Using *k*-means with an initialization equivalent to starting from Goemans–Williamson rounding solutions (K-MeansNM), and keeping track of the points visited by *k*-means at all time, we can guarantee that the approximation guarantee is maintained, with the possibility of finding larger cut values. For the other clustering algorithms, this is not true; however, for one version of *k*-means (K-Means2N), the same or better solutions than RR were found, even without the guarantee. On average, the remaining clustering algorithms yield larger cut values than RR, and the number of instances where those algorithms find lower cuts are less than 15% for the worst case (K-MeansDet), and less than 5% for the others. Our approach is not guaranteed to improve all instances. In particular, it does not result in a theoretical improvement of the Goemans–Williamson approximation guarantee.

We have derived, however, an *instance-specific* lower bound for the approximation ratio that depends both on the instance and the solution, i.e., the cut itself. It provides a plausible measure of performance also for instances with unknown maximal cut value.

The success of *k*-means related clustering approaches suggests to extend this idea to other clustering methods. For spectral clustering, for instance, a natural starting points would be an auxiliary graph with weights $\omega_{ij} = \max(\vec{v}_i \cdot \vec{v}_j)$ and edge set $\{(i,j)|\omega_{ij} > 0\}$.

Author Contributions: All authors jointly contributed the conceptualization of the study, the theoretical results, the interpretation of the benchmark data, and the writing of the manuscript. A.E.R.-F. implemented the software and performed the benchmark. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the *Consejo Nacional de Ciencia y Tecnología* (CONACyT), and the *University of California Institute for Mexico and the United States* (UC MEXUS). The APC was funded by Library of Leipzig University.

Acknowledgments: We thank Giovanni Rinaldi and Laura Palagi for the discussions about the solver of the SDP problem.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the scientific content of this work.

Abbreviations

The following abbreviations are used in this manuscript:

IQP	Integer Quadratic Programming
VP	Vector Programming
SDP	Semidefinite Programming
Fuzzy	Fuzzy c-means clustering
K-MeansRand	Randomized version of k-Means
K-MeansDet	Deterministic version of k-Means
K-MedRand	Randomized version of k-Medoids
K-MedDet	Deterministic version of k-Medoids
MST	Minimum Spanning Tree
RR	Randomized Rounding (Goemans–Williamson rounding)
K-Means2N	Randomized version of k-Means initialized with 2 random vectors
K-MeansNM	Deterministic version of k-Means initialized with a random vector and its negative

References

1. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computation*; Miller, R.E., Thacher, J.W., Eds.; Plenum Press: New York, NY, USA, 1972; pp. 85–103.
2. Papadimitriou, C.H.; Yannakakis, M. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.* **1991**, *43*, 425–440, doi:10.1016/0022-0000(91)90023-X.
3. Hadlock, F. Finding a Maximum Cut of a Planar Graph in Polynomial Time. *SIAM J. Comput.* **1975**, *4*, 221–225, doi:10.1137/0204019.
4. Grötschel, M.; Nemhauser, G.L. A polynomial algorithm for the max-cut problem on graphs without long odd cycles. *Math. Program.* **1984**, *29*, 28–40, doi:10.1007/BF02591727.
5. Bodlaender, H.L.; Jansen, K. On the complexity of the maximum cut problem. *Nord. J. Comput.* **2000**, *7*, 14–31, doi:10.5555/640044.640046.
6. Anglès d'Auriac, J.C.; Preissmann, M.; Sebö, A. Optimal cuts in graphs and statistical mechanics. *Math. Comput. Model.* **1997**, *26*, 1–11, doi:10.1016/S0895-7177(97)00195-7.
7. Festa, P.; Pardalos, P.M.; Resende, M.G.C.; Ribeiro, C.C. Randomized heuristics for the Max-Cut problem. *Optim. Methods Softw.* **2002**, *17*, 1033–1058, doi:10.1080/1055678021000090033.
8. Klemm, K.; Mehta, A.; Stadler, P.F. Landscape Encodings Enhance Optimization. *PLoS ONE* **2012**, *7*, e34780, doi:10.1371/journal.pone.0034780.
9. Ma, F.; Hao, J.K. A multiple search operator heuristic for the max-*k*-cut problem. *Ann. Oper. Res.* **2017**, *248*, 365–403, doi:10.1007/s10479-016-2234-0.
10. Shao, S.; Zhang, D.; Zhang, W. A simple iterative algorithm for maxcut. *arXiv* **2019**, arXiv:1803.06496
11. Delorme, C.; Poljak, S. Laplacian eigenvalues and the maximum cut problem. *Math. Program.* **1993**, *62*, 557–574, doi:10.1007/BF01585184.
12. Poljak, S.; Rendl, F. Solving the max-cut problem using eigenvalues. *Discret. Appl. Math.* **1995**, *62*, 249–278, doi:10.1016/0166-218X(94)00155-7.
13. Trevisan, L. Max Cut and the Smallest Eigenvalue. *SIAM J. Comput.* **2012**, *41*, 1769–1786, doi:10.1137/090773714.

14. Soto, J.A. Improved Analysis of a Max-Cut Algorithm Based on Spectral Partitioning. *SIAM J. Discret. Math.* **2015**, *29*, 259–268, doi:10.1137/14099098X.
15. Goemans, M.X.; Williamson, D.P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **1995**, *42*, 1115–1145, doi:10.1145/227683.227684.
16. Grippo, L.; Palagi, L.; Piccialli, V. An unconstrained minimization method for solving low-rank SDP relaxations of the maxcut problem. *Math. Program.* **2011**, *126*, 119–146, doi:10.1007/s10107-009-0275-8.
17. Palagi, L.; Piccialli, V.; Rendl, F.; Rinaldi, G.; Wiegele, A. Computational Approaches to Max-Cut. In *Handbook on Semidefinite, Conic and Polynomial Optimization*; Springer: Boston, MA, USA, 2011; pp. 821–847, doi:10.1007/978-1-4614-0769-0_28.
18. Mahajan, S.; Ramesh, H. Derandomizing semidefinite programming based approximation algorithms. In Proceedings of the IEEE 36th Annual Foundations of Computer Science, Milwaukee, WI, USA, 23–25 October 1995; pp. 162–169, doi:10.1109/SFCS.1995.492473.
19. Håstad, J. Some optimal inapproximability results. *J. ACM* **2001**, *48*, 798–859, doi:10.1145/502090.502098.
20. Khot, S.; Kindler, G.; Mossel, E.; O'Donnell, R. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.* **2007**, *37*, 319–357, doi:10.1137/S0097539705447372.
21. Feige, U.; Karpinski, M.; Karpinski, M. Improved approximation of Max-Cut on graphs of bounded degree. *J. Algorithms* **2002**, *43*, 201–219, doi:10.1016/S0196-6774(02)00005-6.
22. MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*; University of California Press: Berkeley, CA, USA, 1967; Volume 1, pp. 281–297.
23. Dhillon, I.S.; Modha, D.S. Concept Decompositions for Large Sparse Text Data Using Clustering. *Mach. Learn.* **2001**, *42*, 143–175, doi:10.1023/A:1007612920971.
24. Kaufmann, L.; Rousseeuw, P. Clustering by Means of Medoids. In *Data Analysis Based on the L_1-Norm and Related Methods*; Dodge, Y., Ed.; North-Holland: Amsterdam, The Netherlands, 1987; pp. 405–416.
25. Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Springer: Boston, MA, USA, 1981. doi:10.1007/978-1-4757-0450-1.
26. Grygorash, O.; Zhou, Y.Z.; Jorgensen, Z. Minimum Spanning Tree Based Clustering Algorithms. In Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), Arlington, VA, 13–15 November 2006; Lu, C.T.L., Bourbakis, N.G.B., Eds.; IEEE Computer Society: Los Alamitos, CA, USA, 2006; pp. 73–81, doi:10.1109/ICTAI.2006.83.
27. Bishop, C.M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*; Springer: Berlin/Heidelberg, Germany, 2006.
28. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: Berlin/Heidelberg, Germany, 2009.