Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Geometry 28 (2004) 89–112
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

A local search approximation algorithm for k-means clustering ☆

Tapas Kanungo [a], David M. Mount [b,*,1], Nathan S. Netanyahu [c,d], Christine D. Piatko [e], Ruth Silverman [d], Angela Y. Wu [f]

[a] *IBM Almaden Research Center, San Jose, CA 95120, USA*
[b] *Department of Computer Science, University of Maryland, College Park, MD, USA*
[c] *Department of Mathematics and Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel*
[d] *Center for Automation Research, University of Maryland, College Park, MD, USA*
[e] *The Johns Hopkins University Applied Physics Laboratory, Laurel, MD, USA*
[f] *Department of Computer Science, American University, Washington, DC, USA*

Available online 9 April 2004

Communicated by C. Bajaj

Abstract

In k-means clustering we are given a set of n data points in d-dimensional space $\Re^d$ and an integer k, and the problem is to determine a set of k points in $\Re^d$, called *centers*, to minimize the mean squared distance from each data point to its nearest center. No exact polynomial-time algorithms are known for this problem. Although asymptotically efficient approximation algorithms exist, these algorithms are not practical due to the very high constant factors involved. There are many heuristics that are used in practice, but we know of no bounds on their performance.

We consider the question of whether there exists a simple and practical approximation algorithm for k-means clustering. We present a local improvement heuristic based on swapping centers in and out. We prove that this yields a $(9+\varepsilon)$-approximation algorithm. We present an example showing that any approach based on performing a fixed number of swaps achieves an approximation factor of at least $(9-\varepsilon)$ in all sufficiently high dimensions. Thus, our approximation factor is almost tight for algorithms based on performing a fixed number of swaps. To establish the practical value of the heuristic, we present an empirical study that shows that, when combined with Lloyd's algorithm, this heuristic performs quite well in practice.
© 2004 Elsevier B.V. All rights reserved.

☆ A preliminary version of this paper appeared in the 18th Annual ACM Symposium on Computational Geometry (SoCG'02), June 2002, Barcelona, Spain, 10–18.

* Corresponding author.
E-mail addresses: kanungo@almaden.ibm.com (T. Kanungo), mount@cs.umd.edu (D.M. Mount), nathan@macs.biu.ac.il (N.S. Netanyahu), christine.piatko@jhuapl.edu (C.D. Piatko), ruth@cfar.umd.edu (R. Silverman), awu@american.edu (A.Y. Wu).

[1] This material is based upon work supported by the National Science Foundation under Grant No. 0098151.

0925-7721/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comgeo.2004.03.003

Keywords: Clustering; k-means; Approximation algorithms; Local search; Computational geometry

1. Introduction

Clustering problems arise in many different applications, including data mining and knowledge discovery [15], data compression and vector quantization [19], and pattern recognition and pattern classification [11]. There are many approaches, including splitting and merging methods such as ISODATA [6,21], randomized approaches such as CLARA [25] and CLARANS [34], and methods based on neural nets [27]. Further information on clustering and clustering algorithms can be found in [8,20–23,25]. One of the most popular and widely studied clustering methods for points in Euclidean space is called *k-means clustering*. Given a set P of n *data points* in real d-dimensional space $\Re^d$, and an integer k, the problem is to determine a set of k points in $\Re^d$, called *centers*, to minimize the mean squared Euclidean distance from each data point to its nearest center. This measure is often called the *squared-error distortion* [19,21]. Clustering based on k-means is closely related to a number of other clustering and facility location problems. These include the Euclidean *k-median* [3,28] and the *Weber problem* [42], in which the objective is to minimize the sum of distances to the nearest center, and the Euclidean *k-center problem* [13,39], in which the objective is to minimize the maximum distance. There are no efficient exact solutions known to any of these problems for general k, and some formulations are NP-hard [18].

Given the apparent difficulty of solving the k-means and other clustering and location problems exactly, it is natural to consider approximation, either through polynomial-time approximation algorithms, which provide guarantees on the quality of their results, or heuristics, which make no guarantees. One of the most popular heuristics for the k-means problem is *Lloyd's algorithm* [17,30,31], which is often called the *k-means algorithm*. Define the *neighborhood* of a center point to be the set of data points for which this center is the closest. It is easy to prove that any locally minimal solution must be *centroidal*, meaning that each center lies at the centroid of its neighborhood [10,14]. Lloyd's algorithm starts with any feasible solution, and it repeatedly computes the neighborhood of each center and then moves the center to the centroid of its neighborhood, until some convergence criterion is satisfied. It can be shown that Lloyd's algorithm eventually converges to a locally optimal solution [38]. Computing nearest neighbors is the most expensive step in Lloyd's algorithm, but a number of practical implementations of this algorithm have been discovered recently [2,24,35–37].

Unfortunately, it is easy to construct situations in which Lloyd's algorithm converges to a local minimum that is arbitrarily bad compared to the optimal solution. Such an example is shown in Fig. 1 for $k = 3$ and where $x < y < z$. The optimal distortion is $x^2/4$, but it is easy to verify that the solution shown at the bottom is centroidal and has a distortion of $y^2/4$. By increasing the ratio y/x the approximation ratio for Lloyd's algorithm can be made arbitrarily high. There are many other heuristics for k-means

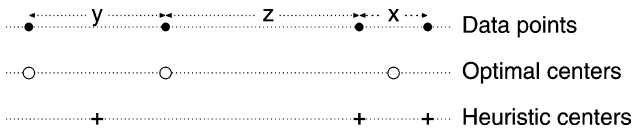

Fig. 1. Lloyd's algorithm can produce an arbitrarily high approximation ratio.

clustering, based on methods such as branch-and-bound searching, gradient descent, simulated annealing, and genetic algorithms [7,12,41]. No proven approximation bounds are known for these methods.

It is desirable to have some bounds on the quality of a heuristic. Given a constant $c \geqslant 1$, a *c-approximation algorithm* (for a minimization problem) produces a solution that is at most a factor c larger than the optimal solution. There is a classical tradeoff between approximation factors and running times. Some clustering algorithms are able to produce solutions that are arbitrarily close to optimal. This includes $(1+\varepsilon)$-approximation algorithms for the Euclidean k-median problem by Arora, Raghavan and Rao [3] and by Kolliopoulos and Rao [28]. The latter achieves a running time of $O(2^{1/\varepsilon^d} n \log n \log k)$, assuming that the dimension d is fixed. It is based on applying dynamic programming to an adaptive hierarchical decomposition of space. Another example is the $(1+\varepsilon)$-approximation algorithm for the Euclidean k-center problem given by Agarwal and Procopiuc, which runs in $O(n \log k) + (k/\varepsilon)^{O(k^{1-1/d})}$ time [1].

Matoušek [32] achieved an important breakthrough by presenting an asymptotically efficient $(1+\varepsilon)$-approximation algorithm for k-means clustering, which runs in $O(n(\log n)^k \varepsilon^{-2k^2 d})$ time for fixed k and d. First, Matoušek shows how to compute a set of $O(n\varepsilon^{-d} \log(1/\varepsilon))$ candidate centers, called an *ε-approximate centroid set*, from which an approximately optimal solution may be drawn. He then shows that a near-optimal solution may be assumed to consist of a *well-spread* k-tuple, which intuitively means that no subset of the k-tuple is strongly isolated relative to the other points. Finally, he proves that given a set of m points, there are $O(m\varepsilon^{-k^2 d})$ such well-spread sets. The algorithm generates all these tuples and returns the k-tuple with the minimum distortion. Unfortunately, the constant factors are well beyond practical ranges unless d and k are very small. In Section 4, we show that, under reasonable assumptions about the way in which the candidate centers are chosen (which Matoušek's algorithm satisfies), the number of well-spread k-tuples that the algorithm generates is at least $(2/\varepsilon)^k$. In typical applications, k may range from tens to hundreds, and so this is well beyond practical limits. The dynamic programming approximation algorithm presented by Kolliopoulos and Rao for the k-median problem [28] is also a candidate for modification, but also suffers from similarly large constant factors.

Another common approach in approximation algorithms is to develop much more practical, efficient algorithms having weaker, but still constant, approximation factors. This includes the work of Thorup on solving location problems in sparse graphs [40] and by Mettu and Plaxton [33] on the use of successive swapping for the metric k-means problem. The most closely related work to our own are the recent approximation algorithms for the metric k-median problem by Korupolu, Plaxton and Rajaraman [29], Charikar and Guha [9], and Arya et al. [5]. These algorithms are based on *local search*, that is, by incrementally improving a feasible solution by swapping a small number of points in and out of the solution set.

In this paper we present such an approximation algorithm for k-means based on a simple swapping process. In Section 2 we derive an approximation ratio of $9+\varepsilon$ for the heuristic. Our approach is based on the heuristic for k-medians presented by Arya et al. [5]. However, due to the different nature of the k-means problem, the analysis is different and relies on geometric properties that are particular to the k-means problem. In Section 3 we show that this bound is essentially tight for the class of local search algorithms that are based on performing a constant number of swaps. In particular, we present an example showing that any approach based on performing a fixed number of swaps cannot achieve an approximation factor of better than $(9-\varepsilon)$ in all sufficiently high dimensions.

Approximation factors as high as 9 are of little practical value. Nonetheless, we believe that a combination of local search and existing approaches results in a practical approximation algorithm with performance guarantees. In Section 5 we present a hybrid approximation algorithm based on combining local search with Lloyd's algorithm. We provide empirical evidence that this hybrid algorithm provides results that are as good or better than Lloyd's algorithm, both in terms of distortion and running time.

2. The local search algorithm

Given $u, v \in \Re^d$, let $\Delta(u, v)$ denote the squared Euclidean distance between these points, that is

$$\Delta(u, v) = \text{dist}^2(u, v) = \sum_{i=1}^{d} (u_i - v_i)^2 = (u - v) \cdot (u - v),$$

where $u \cdot v$ denotes the dot product of vectors u and v. Given a finite set $S \subset \Re^d$, define its *distortion* relative to any point v to be $\Delta(S, v) = \sum_{u \in S} \Delta(u, v)$.

Consider a set P of n *data points* in $\Re^d$ and an integer k. Given any set S of k points, for any $q \in \Re^d$ define s_q to be the closest point of S to q. Our goal is to compute the k-element point set S that minimizes the total *distortion* of S relative to P, defined as

$$\Delta_P(S) = \sum_{q \in P} \Delta(q, s_q).$$

When P is understood, we will refer to this simply as $\Delta(S)$.

The principal difficulty in extending existing approaches for the metric k-medians problem to k-means is that squared distances do not define a metric, and in particular they do not satisfy the triangle inequality, which states that for any points u, v and w, $\text{dist}(u, w) \leqslant \text{dist}(u, v) + \text{dist}(v, w)$. When considering squared distances we have

$$\begin{aligned}\Delta(u, w) &\leqslant \big(\text{dist}(u, v) + \text{dist}(v, w)\big)^2 \\ &= \text{dist}^2(u, v) + 2\,\text{dist}(u, v)\,\text{dist}(v, w) + \text{dist}^2(v, w) \\ &\leqslant \Delta(u, v) + \Delta(v, w) + 2\,\text{dist}(u, v)\,\text{dist}(v, w).\end{aligned}$$

The final product term can be bounded by observing that $2ab \leqslant a^2 + b^2$, for any a and b. Hence we have the following *doubled triangle inequality*:

$$\Delta(u, w) \leqslant 2\big(\Delta(u, v) + \Delta(v, w)\big).$$

One obvious idea for producing a local improvement heuristic for k-means would be to generalize the methods of Arya et al. [5] for the metric k-median problem using this doubled triangle inequality. Unfortunately, this does not seem to work because their analysis relies crucially on the triangle inequality. In particular, a cancellation of terms that arises in their analysis fails to hold when the triangle inequality is doubled.

Our approach is based on two ideas. The first is the introduction of an alternative to the triangle inequality, which, unlike the doubled triangle inequality is sensitive to the ratio of the optimal and heuristic solution (see Lemma 2.3 below). The second is based on the well-known fact that the optimal solution is centroidal (see [10]). Let $N_S(s)$ denote the neighborhood of s, that is, the set of data points

that are closer to s than to any point in S. By treating points as vectors, the centroidal property implies that

$$s = \frac{1}{|N_S(s)|} \sum_{u \in N_S(s)} u.$$

An important property of centroidal solutions is presented in the following lemma. It states that for the purposes of computing distortions, a set of points may be treated like a point mass centered about its centroid. It follows from a straightforward manipulation of the definition of distortion, but we include the proof for completeness.

Lemma 2.1. *Given a finite subset S of points in $\Re^d$, let c be the centroid of S. Then for any $c' \in \Re^d$, $\Delta(S, c') = \Delta(S, c) + |S|\Delta(c, c')$.*

Proof. By expanding the definition of $\Delta(S, c')$ we have

$$\begin{aligned}
\Delta(S, c') &= \sum_{u \in S} \Delta(u, c') = \sum_{u \in S} (u - c') \cdot (u - c') \\
&= \sum_{u \in S} \big((u - c) + (c - c')\big) \cdot \big((u - c) + (c - c')\big) \\
&= \sum_{u \in S} \big((u - c) \cdot (u - c)\big) + 2\big((u - c) \cdot (c - c')\big) + \big((c - c') \cdot (c - c')\big) \\
&= \Delta(S, c) + 2\left((c - c') \cdot \sum_{u \in S} (u - c)\right) + |S|\big((c - c') \cdot (c - c')\big) \\
&= \Delta(S, c) + |S|\Delta(c, c').
\end{aligned}$$

The last step follows from the fact that if c is S's centroid then $\sum_{u \in S}(u - c)$ is the zero vector. □

2.1. The single-swap heuristic

To illustrate our method, we first present a simple local search that provides a $(25 + \varepsilon)$-approximation to the k-means problem. Our approach is similar to approaches used in other local search heuristics for facility location and k-medians by Charikar and Guha [9] and Arya et al. [5].

In the statement of the k-means problem, the centers may be placed anywhere in space. In order to apply our local improvement search, we need to assume that we are given a discrete set of *candidate centers* C from which k centers may be chosen. As mentioned above, Matoušek [32] showed that C may be taken to be an ε-approximate centroid set of size $O(n\varepsilon^{-d} \log(1/\varepsilon))$, which can be computed in time $O(n \log n + n\varepsilon^{-d} \log(1/\varepsilon))$. Henceforth, when we use the term "optimal", we mean the k-element subset of C having the lowest distortion.

This *single-swap heuristic* operates by selecting an initial set of k centers S from the candidate centers C, and then it repeatedly attempts to improve the solution by removing one center $s \in S$ and replacing it with another center $s' \in C - S$. Let $S' = S - \{s\} \cup \{s'\}$ be the new set of centers. If the modified solution has lower distortion, then S' replaces S, and otherwise S is unchanged. In practice this process is repeated until some long consecutive run of swaps have been performed with no significant decrease in the distortion. By extension of standard results [5,9] it can be shown that by sacrificing a

small factor $\varepsilon > 0$ in the approximation ratio, we can guarantee that this procedure converges after a polynomial number of swaps.

For simplicity, we will assume that the algorithm terminates when no single swap results in a decrease in distortion. Such a set of centers is said to be 1-*stable*. Letting O denote an optimal set of k centers, a set S of k centers is 1-stable then we have

$$\Delta\big(S - \{s\} \cup \{o\}\big) \geqslant \Delta(S) \quad \text{for all } s \in S,\ o \in O. \tag{1}$$

(In fact this is true no matter what O is, but our analysis only relies on this weaker property.) Using this along with the fact that the optimal solution is centroidal, we will establish the main result of this section, which is stated below.

Theorem 2.1. *Let S denote a* 1*-stable set of k centers, and let O denote the optimal set of k centers. Then $\Delta(S) \leqslant 25\Delta(O)$.*

Note that the actual approximation bound is larger by some $\varepsilon > 0$, due to the errors induced by using a discrete set of candidate centers C and the approximate convergence criterion described above. Our analysis is similar in structure to that given by Arya et al. [5], but there are two significant differences. The first is that our notion of capturing a center is different from theirs, and is based on the distance to the closest center, rather than on the numbers of data points assigned to a center. The second is that their permutation function π is not needed in our case, and instead we rely on the centroidal properties of the optimal solution.

For each optimal center $o \in O$, let s_o denote its closest heuristic center in S. We say that o is *captured* by s_o. Note that each optimal center is captured by exactly one heuristic center, but each heuristic center may capture any number of optimal centers. We say that a heuristic center is *lonely* if it captures no optimal center. The analysis is based on constructing a set of *swap pairs*, considering the total change in distortion that results, and then apply Eq. (1) above to bound the overall change in distortion.

We begin by defining a simultaneous partition of the heuristic centers and optimal centers into two sets of groups $S_1, S_2, \ldots, S_r$ and $O_1, O_2, \ldots, O_r$ for some r, such that $|S_i| = |O_i|$ for all i. For each heuristic center s that captures some number $m \geqslant 1$ of optimal centers, we form a group of m optimal centers consisting of these captured centers. The corresponding group of heuristic centers consists of s together with any $m - 1$ lonely heuristic centers. (See Fig. 2.)

We generate the swap pairs as follows. For every partition that involves one captured center we generate a swap pair consisting of the heuristic center and its captured center. For every partition containing two or more captured centers we generate swap pairs between the lonely heuristic centers

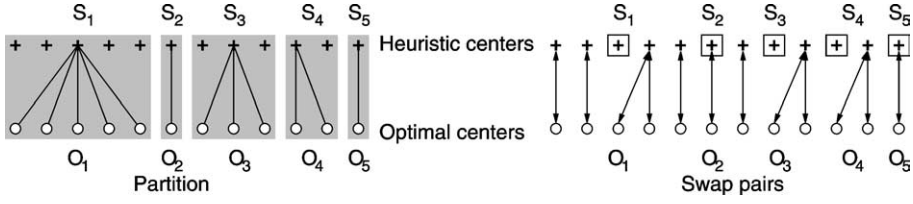

Fig. 2. Partitioning of the heuristic and optimal centers for analysis and the swap pairs. On the left, edges represent the capturing relation, and on the right they represent swap pairs.

and the optimal centers, so that each optimal center is involved in exactly one swap pair and each lonely center is involved in at most two swap pairs. It is easy to verify that:

(1) each optimal center is swapped in exactly once,
(2) each heuristic center is swapped out at most twice, and
(3) if s and o are swapped, then s does not capture any optimal center other than o.

We establish an upper bound on the change in distortion resulting from any such swap pair $\langle s, o\rangle$ by prescribing a feasible (but not necessarily optimal) assignment of data points to the centers $S - \{s\} \cup \{o\}$. First, the data points in $N_O(o)$ are assigned to o, implying a change in distortion of

$$\sum_{q \in N_O(o)} \big(\Delta(q, o) - \Delta(q, s_q)\big). \tag{2}$$

Each point $q \in N_S(s) \setminus N_O(o)$ has lost s as a center and must be *reassigned* to a new center. Let o_q denote the closest optimal center to q. Since q is not in $N_O(o)$ we know that $o_q \neq o$, and hence by property (3) above s does not capture o_q. Therefore, s_{o_q}, the nearest heuristic center to o_q, exists after the swap. We assign q to s_{o_q}. Thus the change in distortion due to this reassignment is at most

$$\sum_{q \in N_S(s) \setminus N_O(o)} \big(\Delta(q, s_{o_q}) - \Delta(q, s)\big). \tag{3}$$

By combining over all swap pairs the change in distortion due to optimal assignment and reassignment together with Eq. (1) we obtain the following.

Lemma 2.2. *Let S be a 1-stable set of k centers, and let O be an optimal set of k centers, then*

$$0 \leqslant \Delta(O) - 3\Delta(S) + 2R,$$

where $R = \sum_{q \in P} \Delta(q, s_{o_q})$.

Proof. Consider just the swap pair $\langle s, o\rangle$. By Eqs. (2) and (3) and the fact that S is 1-stable we have

$$\sum_{q \in N_O(o)} \big(\Delta(q, o) - \Delta(q, s_q)\big) + \sum_{q \in N_S(s) \setminus N_O(o)} \big(\Delta(q, s_{o_q}) - \Delta(q, s)\big) \geqslant 0.$$

To bound the sum over all swap pairs, we recall that each optimal center is swapped in exactly once, and hence each point q contributes once to the first sum. Note that the quantity in the second sum is always nonnegative (because $s_{o_q} \in S$ and s is the closest center in S to q). Hence by extending the sum to all $q \in N_S(s)$ we can only increase its value. Recalling that each heuristic center is swapped in at most twice we have

$$\begin{aligned}
0 &\leqslant \sum_{q \in P} \big(\Delta(q, o_q) - \Delta(q, s_q)\big) + 2 \sum_{q \in P} \big(\Delta(q, s_{o_q}) - \Delta(q, s_q)\big), \\
0 &\leqslant \sum_{q \in P} \Delta(q, o_q) - 3 \sum_{q \in P} \Delta(q, s_q) + 2 \sum_{q \in P} \Delta(q, s_{o_q}), \\
0 &\leqslant \Delta(O) - 3\Delta(S) + 2R,
\end{aligned}$$

from which the desired conclusion follows. □

The term R above is called the *total reassignment cost*. By applying Lemma 2.1 to each optimal neighborhood, we have

$$\begin{aligned} R &= \sum_{o\in O}\sum_{q\in N_O(o)} \Delta(q,s_o) = \sum_{o\in O} \Delta\big(N_O(o),s_o\big) \\ &= \sum_{o\in O}\big(\Delta\big(N_O(o),o\big) + \big|N_o(O)\big|\Delta(o,s_o)\big) = \sum_{o\in O}\sum_{q\in N_O(o)} \big(\Delta(q,o)+\Delta(o,s_o)\big). \end{aligned}$$

Because s_o is the closest heuristic center to o, for each $q\in N_O(o)$, we have $\Delta(o,s_o)\leqslant\Delta(o,s_q)$. This yields

$$R \leqslant \sum_{o\in O}\sum_{q\in N_O(o)} \big(\Delta(q,o)+\Delta(o,s_q)\big) = \sum_{q\in P}\big(\Delta(q,o_q)+\Delta(o_q,s_q)\big).$$

By applying the triangle inequality and expanding we obtain

$$\begin{aligned} R &\leqslant \sum_{q\in P}\Delta(q,o_q) + \sum_{q\in P}\big(\operatorname{dist}(o_q,q)+\operatorname{dist}(q,s_q)\big)^2 \\ &= \sum_{q\in P}\Delta(q,o_q) + \sum_{q\in P}\big(\operatorname{dist}^2(o_q,q)+2\operatorname{dist}(o_q,q)\operatorname{dist}(q,s_q)+\operatorname{dist}^2(q,s_q)\big) \\ &= 2\sum_{q\in P}\Delta(q,o_q) + \sum_{q\in P}\Delta(q,s_q) + 2\sum_{q\in P}\operatorname{dist}(q,o_q)\operatorname{dist}(q,s_q) \\ &= 2\Delta(O)+\Delta(S)+2\sum_{q\in P}\operatorname{dist}(q,o_q)\operatorname{dist}(q,s_q). \end{aligned}$$

To bound the last term we will apply the following technical lemma.

Lemma 2.3. *Let $\langle o_i\rangle$ and $\langle s_i\rangle$ be two sequences of reals, such that $\alpha^2 = (\sum_i s_i^2)/(\sum_i o_i^2)$, for some $\alpha>0$. Then*

$$\sum_{i=1}^n o_i s_i \leqslant \frac{1}{\alpha}\sum_{i=1}^n s_i^2.$$

Proof. By Schwarz's inequality [16] we have

$$\sum_{i=1}^n o_i s_i \leqslant \left(\sum_{i=1}^n o_i^2\right)^{1/2}\left(\sum_{i=1}^n s_i^2\right)^{1/2} = \left(\frac{1}{\alpha^2}\sum_{i=1}^n s_i^2\right)^{1/2}\left(\sum_{i=1}^n s_i^2\right)^{1/2} = \frac{1}{\alpha}\sum_{i=1}^n s_i^2,$$

as desired. □

To complete the analysis, let the o_i sequence consist of $\operatorname{dist}(q,o_q)$ over all $q\in P$, and let the s_i sequence consist of $\operatorname{dist}(q,s_q)$. Let α denote the square root of the approximation ratio, so that

$$\alpha^2 = \frac{\Delta(S)}{\Delta(O)} = \frac{\sum_{q\in P}\operatorname{dist}^2(q,s_q)}{\sum_{q\in P}\operatorname{dist}^2(q,o_q)} = \frac{\sum_{i=1}^n s_i^2}{\sum_{i=1}^n o_i^2}.$$

By applying Lemma 2.3 we have

$$R \leqslant 2\Delta(O) + \Delta(S) + \frac{2}{\alpha}\sum_{q\in P}\mathrm{dist}^2(q, s_q) = 2\Delta(O) + \Delta(S) + \frac{2}{\alpha}\Delta(S) = 2\Delta(O) + \left(1 + \frac{2}{\alpha}\right)\Delta(S).$$

Now we combine this with Lemma 2.2, yielding

$$\begin{aligned} 0 &\leqslant \Delta(O) - 3\Delta(S) + 2\left(2\Delta(O) + \left(1 + \frac{2}{\alpha}\right)\Delta(S)\right) \\ &\leqslant 5\Delta(O) - \left(1 - \frac{4}{\alpha}\right)\Delta(S). \end{aligned} \tag{4}$$

Through simple rearrangements we can express this in terms of α alone:

$$\begin{aligned} &\frac{5}{1 - 4/\alpha} \geqslant \frac{\Delta(S)}{\Delta(O)} = \alpha^2, \\ &5 \geqslant \alpha^2\left(1 - \frac{4}{\alpha}\right), \\ &0 \geqslant \alpha^2 - 4\alpha - 5 = (\alpha - 5)(\alpha + 1). \end{aligned}$$

This implies that $\alpha \leqslant 5$, and hence the approximation ratio of the simple heuristic is bounded by $\alpha^2 \leqslant 25$. This completes the proof of Theorem 2.1.

2.2. The multiple-swap heuristic

We generalize the single-swap approach to provide a factor $9 + \varepsilon$ approximation ratio. Rather than swapping a single pair of points at any time, for some integer p, we consider simultaneous swaps between any subset of S of size $p' \leqslant p$ with any p'-element subset of candidate centers. Otherwise the algorithm is the same. We say that a set of centers is *p-stable* if no simultaneous swap of p elements decreases the distortion. Our main result is given below. As before, there is an additional ε term in the final error because of the use of the discrete candidate centers and the approximate convergence conditions.

Theorem 2.2. *Let S denote a p-stable set of k centers, and let O denote the optimal set of k centers. Then $\Delta(S) \leqslant (3 + \frac{2}{p})^2\Delta(O)$.*

Again our approach is similar to that of Arya et al. [5], but using our different notion of capturing. We define our swaps as follows. Recall the simultaneous partitions of heuristic and optimal centers used in the simple heuristic. If for some i, $|S_i| = |O_i| \leqslant p$, then we create a simultaneous swap involving the sets S_i and O_i. Otherwise, if $|S_i| = |O_i| = m > p$, then for each of the $m - 1$ lonely centers of S_i we generate individual 1-for-1 swaps with all m optimal centers of O_i. For the purposes of the analysis, the change in distortion due to each of these 1-for-1 swaps is weighted by a multiplicative factor of $1/(m - 1)$. (For example, Fig. 3 shows the swaps that would result from Fig. 2 for $p = 3$. The swaps appearing in shaded boxes are performed simultaneously. The 1-for-1 swaps performed between S_1 and O_1 are each weighted by $1/4$.)

It is easy to verify that: (1) each optimal center is swapped in with total weight 1, (2) each heuristic center is swapped out with weight at most $1 + 1/p$, and (3) if sets S' and O' are swapped, then S' captures no optimal centers outside of O'.

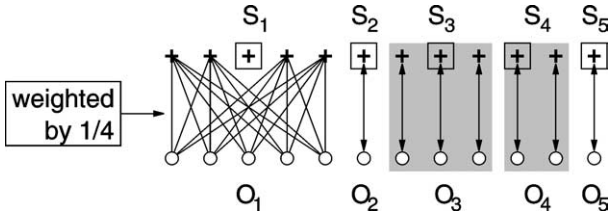

Fig. 3. Swaps for $p = 3$. Shaded regions indicate swaps that are performed simultaneously.

The analysis proceeds in the same manner as the simple case. Because of the replacement of the factor 2 with $(1 + 1/p)$, the inequalities in the proof of Lemma 2.2 now become

$$0 \leqslant \sum_{q\in P}\bigl(\Delta(q, o_q) - \Delta(q, s_q)\bigr) + \left(1 + \frac{1}{p}\right)\sum_{q\in P}\bigl(\Delta(q, s_{o_q}) - \Delta(q, s_q)\bigr),$$

$$0 \leqslant \Delta(O) - \left(2 + \frac{1}{p}\right)\Delta(S) + \left(1 + \frac{1}{p}\right)R.$$

The analysis and the definition of α proceed as before, and Eq. (4) becomes

$$\begin{aligned} 0 &\leqslant \Delta(O) - \left(2 + \frac{1}{p}\right)\Delta(S) + \left(1 + \frac{1}{p}\right)\left(2\Delta(O) + \left(1 + \frac{2}{\alpha}\right)\Delta(S)\right) \\ &\leqslant \left(3 + \frac{2}{p}\right)\Delta(O) - \left(1 - \frac{2}{\alpha}\left(1 + \frac{1}{p}\right)\right)\Delta(S). \end{aligned}$$

Again, by rearranging and expressing in terms of α we have

$$\frac{3 + (2/p)}{1 - (2/\alpha)(1 + 1/p)} \geqslant \frac{\Delta(S)}{\Delta(O)} = \alpha^2,$$

$$0 \geqslant \alpha^2 - 2\alpha\left(1 + \frac{1}{p}\right) - \left(3 + \frac{2}{p}\right) \geqslant \left(\alpha - \left(3 + \frac{2}{p}\right)\right)(\alpha + 1).$$

This implies that $\alpha \leqslant 3 + 2/p$, and hence the approximation ratio of the general heuristic is α^2, which approaches 9 as p increases.

3. A tight example

It is natural to ask whether the factor 9 is the correct approximation factor for swap-based heuristics, or whether it arises from some slackness in our analysis. In this section we provide evidence that this is probably close to the correct factor assuming an algorithm based on performing a fixed number of swaps. We show that for any p, there is a configuration of points in a sufficiently high dimensional space such that the p-swap heuristic achieves a distortion that is $(9 - \varepsilon)$ times optimal. This example has the nice property that it is centroidal. This implies that it is also a local minimum for Lloyd's algorithm. Hence neither the swap heuristic (assuming swaps with optimal centers) nor Lloyd's algorithm would be able to make further progress. We make the assumption that centers may only be placed at a given discrete set of candidate locations. This candidate set is reasonable in that it contains an ε-approximately optimal

solution. Overcoming this assumption would imply that the entire analysis method would somehow need to be generalized to handle swaps with points other than the optimal centers.

Arya et al. [5] presented a tight example for their heuristic in a metric space. However, their example cannot be embedded in Euclidean space of any dimension and does not even allow centers to be placed at data points. Our approach is quite different.

Theorem 3.1. *Given p and $\varepsilon > 0$, there exists an integer k, a dimension d, a finite set of points $P \in \Re^d$, a finite set of candidate centers C, and a set $S \subseteq C$ of k centers, such that the following hold*:

(i) *C contains an ε-approximately optimal solution.*
(ii) *S is p-stable.*
(iii) *$\Delta(S) \geqslant (9-\varepsilon)\Delta(O)$, where O is the optimal k-means solution.*

In the rest of this section we provide a proof of this theorem. Let d (dimension) and N be even integer parameters to be specified later. Our framework consists of a large d-dimensional integer grid, $G = \{0, 1, \ldots, N-1\}^d$. To avoid messy boundary issues, we may assume that the grid is a topological d-dimensional torus, by taking indices modulo N. For N sufficiently large, this torus may be embedded in $(d+1)$-space, so that distances from each embedded grid point to the embedded image of its grid neighbors are arbitrarily close to 1. Thus the local neighborhoods of all the grid points are identical.

The grid points are d-dimensional integer vectors, where each coordinate is in $\{0, 1, \ldots, N-1\}$. The points of G are labeled even or odd, depending on the parity of the sum of coordinates. Consider a parameter x, $0 < x < 1/2$, to be fixed later. Let $T(x)$ be the following set of $2d$ points displaced at a distance $+x$ and $-x$ from the origin along each of the coordinate axes:

$$(\pm x, 0, 0, \ldots, 0), (0, \pm x, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, \pm x).$$

The data set P consists of the union of translates of $T(x)$ each centered about an even grid point. (See Fig. 4.) Thus, $n = dN^d$. We set $k = n/(2d)$. It is easy to see that the optimal solution O consists of k centers placed at the even grid points. The neighborhood of each center of O consists of $2d$ points, each at distance x. Consider a solution S consisting of k points placed at each of the odd grid points. The neighborhood of each point of S consists of $2d$ points at distance $1-x$.

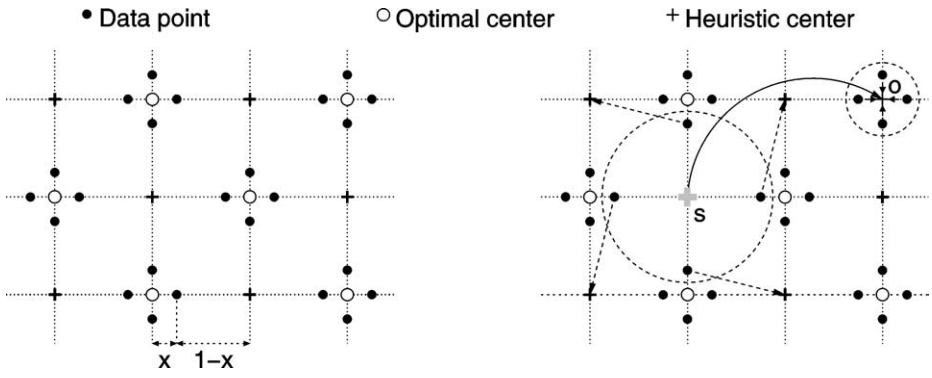

Fig. 4. Example of the lower bound in the plane. Black circles are the data points, hollow circles denote the optimal centers, and crosses denote the heuristic centers.

Each optimal center has a neighborhood of $2d$ points at distance x, and each heuristic center has a neighborhood of $2d$ points at distance $(1-x)$. Thus we have

$$\frac{\Delta(S)}{\Delta(O)} = \frac{(1-x)^2}{x^2}.$$

We argue below that by choosing $x = 1/(4-p/d)$, no p-swap involving points of S and C can improve the distortion. By making d sufficiently large relative to p, this implies that the approximation ratio is arbitrarily close to $(3/4)^2/(1/4)^2 = 9$, as desired.

To show that no p-way swap improves the distortion, consider any simultaneous swap between two p-element subsets S' and O' of heuristic and optimal centers, respectively. Because the optimal neighborhoods are disjoint and each contains $2d$ points, the change in distortion due to assigning these points to their new optimal center is

$$2dp\big(x^2 - (1-x)^2\big) = 2dp(2x-1).$$

No other points are assigned to a closer center.

Now consider the $2dp$ neighbors of heuristic centers that have now been removed. These data points must be reassigned to the nearest existing center. After performing the swap, there are at most p^2 pairs $\langle s, o\rangle$, where $s \in S$ and $o \in O$, such that s and o are adjacent to each other in the grid. For these points no additional reassignment is needed because the point has been moved to its optimal center. For the remaining neighbors of the heuristic centers, of which there are at least $2dp - p^2$, we need to reassign each to a new center. The closest such center is at distance $\sqrt{1+x^2}$. Hence the change in distortion due to reassignment is at least

$$(2dp - p^2)\big((1+x^2) - (1-x)^2\big) = 2dp\left(1 - \frac{p}{2d}\right)2x.$$

Combining these two, the total change in distortion is at least

$$2dp\left(2x - 1 + \left(1 - \frac{p}{2d}\right)2x\right) = 2dp\left(\left(2 - \frac{p}{2d}\right)2x - 1\right).$$

This is nonnegative if we set $x = 1/(4-p/d)$, and hence the p-swap heuristic cannot make progress on this example. This establishes Theorem 3.1.

4. Analysis of well-spread k-tuples

In the introduction we pointed out that Matoušek presented an asymptotically efficient ε-approximation to the k-means problem, under the assumption that k, d and ε are fixed constants [32]. Although this is a very important theoretical result, the constant factors arising in this algorithm are too large to be of practical value, unless k is very small. This raises the question of whether these large constant factors are merely an artifact of the analysis, or whether they are, in some sense, an inescapable consequence of the approach. In this section, we will argue that the latter is the case.

Let us begin with an overview of the essential elements of Matoušek's algorithm. First, recall that a set of candidate centers is called an *ε-approximate centroid set* if, by restricting the selection of centers to this set, the average distortion is larger by a factor of at most $(1+\varepsilon)$. Matoušek shows that given n

points in $\Re^d$, such a set of size $m = O(n\varepsilon^{-d}\log(1/\varepsilon))$ can be computed efficiently. Given such a set the algorithm proceeds by selecting a judicious subset of k-tuples from these candidate points, and argues that one of these subsets provides the desired approximate solution to the k-means problem. Given a real number r and two point sets $Y \subset X$, the set Y is *r-isolated* in X if every point in $X \setminus Y$ is at distance at least $r \cdot \mathrm{diam}(Y)$ from Y. A set X is *ε-well-spread* if there is no proper subset of X of two or more points that is $(1/\varepsilon)$-isolated in X. Matoušek shows that, given a set of m points in $\Re^d$, an ε-well-spread set of k-tuples of size $O(m\varepsilon^{-k^2 d})$ can be computed efficiently, and that restricting attention to such k-tuples produces an ε-approximation. Applying this procedure to the set of candidate points produces the desired approximation.

Of course, the constant factors suggested above are well beyond the bounds of practicality, but might a smaller set suffice? We will prove a lower bound on the number of well-spread k-tuples that would need to be generated in order to guarantee a relative error of ε. Our analysis is based on a *locality assumption* that the choice of candidate centers is based only on the local distribution of the points, and has no knowledge of which cluster each point belongs to in an optimal clustering. This assumption is satisfied by any reasonable selection algorithm, including Matoušek's algorithm.

Theorem 4.1. *There exists a configuration of points in the plane, such that if $\varepsilon \leqslant 1/(3\sqrt{k})$, the number of well-spread k-tuples that need to be tested by Matoušek's algorithm is at least $(2/\varepsilon)^k$.*

Our approach is to present a configuration of points in the plane and argue that, in order to achieve a relative error that is less than ε, the points of the candidate centroid set must be sampled with a certain minimum density. This in turn provides a lower bound on the size of the candidate centroid set, and on the number of well-spread k-tuples.

Our analysis assumes that k is a perfect square and that the points lie in a 2-dimensional square domain of size $\sqrt{k} \times \sqrt{k}$, which is subdivided into a grid of k pairwise disjoint unit squares. (See Fig. 5(a).) Points are distributed identically within each of these squares. Consider any unit square. Let T be a closed square of side length $1/7$ centered at the midpoint of the unit square. (The factor $1/7$ is rather arbitrary, and only affects the constant factor in the analysis. We assume this value is independent of ε.) The points of this unit square are placed uniformly along the boundary of a square S of side length $5/7$ that is centered at an arbitrary point of z within T. (See Fig. 5(b).) It is easy to see that for large n, the optimal k-means solution involves placing one center in each unit square at the center point z.

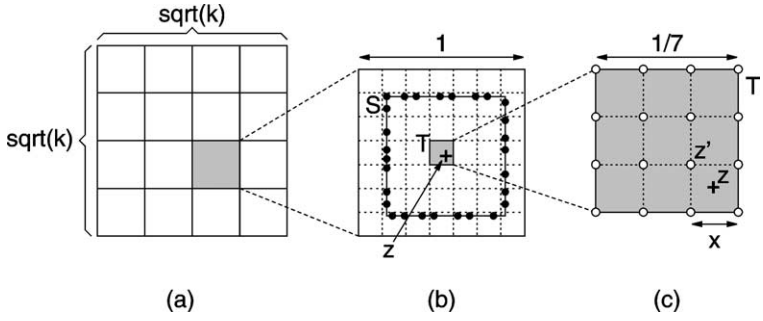

Fig. 5. Analysis of the number of well-spread k-tuples.

For the purposes of producing a lower bound, it suffices to limit consideration to candidate points lying within T. By our locality assumption, the candidate selection algorithm cannot know the location of the optimum center, and, since the distribution of points surrounding T looks about the same to every point of T, the candidate selection process can do no better than select points uniformly throughout T. Let us assume that the candidate points C are taken to be the vertices of a square grid of side length x, where the value of x will be derived below. See Fig. 5(c). (The exact pattern of candidates is not important for our proof, only the assumption of uniformity.) By adjusting the location of z within T, we can place z so that the closest candidate center z' to z is at a squared distance of $2(x/2)^2 = x^2/2$ from z. By applying Lemma 2.1 (where z plays the role of c, and z' plays the role of c'), it follows that the absolute increase in the average squared distortion is equal to the squared distance between z and z', which is $x^2/2$. To derive the relative error, we first need to compute the expected optimal average distortion. Since the points are uniformly distributed along S's boundary, and assuming that n is large, we can estimate this by integrating the squared distance from each point on the boundary of S to the center of S. Straightforward calculations show this to be $(4/3)(5/14)^2 \leqslant 1/4$. Therefore, in order to achieve a relative approximation error of ε, we require that $x^2/2 \leqslant \varepsilon/4$, that is, $x \leqslant \sqrt{\varepsilon/2}$. From this it follows that the number of candidate points in T must be at least $1/x^2 = 2/\varepsilon$. (This lower bound is much smaller than Matoušek's upper bound because our assumption that points are uniformly distributed allows us to ignore n altogether.)

Now, consider any k-tuple formed by selecting any one candidate from each of the candidate sets of the unit squares. We claim that each such set is ε-well-spread, for all sufficiently small ε. The closest that two candidate points can be is $6/7$, and the farthest they can be is at most $2\sqrt{k}$. Thus any subset of two or more points has a diameter of at least $6/7$, and the next closest point is at most a distance of $2\sqrt{k}$ away. It follows that if $(2\sqrt{k}) \leqslant 6/(7\varepsilon)$, any such k-tuple is ε-well-spread. This is satisfied given our hypothesis that $\varepsilon \leqslant 1/(3\sqrt{k})$. Thus, the number of such k-tuples that the algorithm needs to test, in order to guarantee an ε relative error in the average distortion for this 2-dimensional example is at least $(2/\varepsilon)^k$. This completes the proof.

5. Experimental results

Given the relatively high approximation factors involved and the tight example, an important question is whether the swap-based heuristics perform well enough to be of practical value. In this section we argue that indeed these heuristics can be of significant value, especially if applied in conjunction with a locally optimal heuristic such as Lloyd's algorithm.

It is quite easy to see why such a merger is profitable. As mentioned earlier, Lloyd's can get stuck in local minima. One common approach for dealing with this is to run this algorithm repeatedly with different random starting sets. In contrast, the swap heuristic is capable of moving out of a local minimum, but it may take very long to move near to a local minimum. By alternating between the two methods, we have a simple heuristic that takes advantage of both methods' strengths. This is similar in spirit to methods based on simulated annealing [26], but without the complexities of defining temperature schedules and with the advantage of provable performance guarantees.

Our implementation of the swap heuristic differs from the description in this paper in a couple of respects. First, we sampled pairs for swapping randomly, rather than applying some systematic enumeration. This allows the heuristic to be terminated at any point. Also, rather than performing p swaps simultaneously, our heuristic performs swaps one by one. After each individual swap, we compute

the change in distortion. If the distortion decreases after any one swap, we stop immediately, without completing the full sequence of p swaps. This was done so that any improvement that arises from a swap is not undone by a subsequent swap.

One other difference involves the selection of candidate centers. We did not explicitly construct an ε-approximate centroid set as in Matoušek's algorithm [32]. Since the size of such a set is $O(\varepsilon^{-d} n \log(1/\varepsilon))$, storing such a set in higher dimensions is not practical. Instead, we implemented a procedure that is designed to simulate Matoušek's scheme but samples candidate centers on demand. The original points are stored in a kd-tree, in which each leaf node contains one data point. Each node of the tree is associated with an axis-aligned hyper-rectangle, called its *cell*, which contains all the descendent data points. We generate a node of the tree at random. If this is a leaf node, we sample the associated point that is stored in this node. If this is an internal node, we consider the factor-3 expansion of its cell, and sample a point uniformly at random from this expanded cell. In this way, about half the candidate points are sampled randomly from the data set (when a leaf node is sampled), and otherwise they are just points in $\Re^d$.

For purposes of comparison, we also implemented a common variant of Lloyd's algorithm, called *iterated Lloyd's*. In this heuristic, centers are chosen randomly, and some number of stages of Lloyd's algorithm are performed. Recall that each stage consists of computing the neighborhood of each center point, and then moving each center point to the centroid of its neighborhood. Stages are repeated until the following convergence condition is satisfied: over three consecutive stages, the average distortion decreases by less than 10%. We call such a sequence of stages a *run*. After each run, a new random set of centers is generated and the process is repeated until the total number of stages exceeds a prespecified bound. The centers producing the best distortion are saved.

Finally, we implemented a *hybrid heuristic*, which is combination of the swap heuristic with iterated Lloyd's algorithm. This heuristic augments the swap step by first applying one step of the swap heuristic and then follows this with one run of Lloyd's algorithm, as described in the previous paragraph.

The programs were written in C++, compiled with g++, and run on a Sun Ultra 5 workstation. We considered the following two synthetic distributions.

ClusGauss: The data consist of $n = 10{,}000$ points in $\Re^3$, which were generated from a distribution consisting of k clusters of roughly equal sizes, with centers uniformly distributed in a cube of side length 2. The points of each cluster are drawn from a multivariate Gaussian distribution centered at the cluster center, where each coordinate has a given standard deviation σ. We considered $k \in \{25, 50, 100\}$, and $\sigma = 0.05$.

MultiClus: The data consist of $n = 10{,}000$ points in $\Re^3$, which were generated from a distribution consisting of k multivariate Gaussian clusters of various sizes and standard deviations. Again cluster centers were sampled uniformly from a cube of side length 2. The cluster sizes are powers of 2. The probability of generating a cluster of size 2^i is $1/2^i$. The coordinate standard deviation for a cluster of size m is $0.05/\sqrt{m}$, implying that each cluster has roughly the same total distortion. We considered $k \in \{50, 100, 500\}$.

The MultiClus distribution was designed to be a adversary for clustering methods based on simple random sampling. Because most of the points belong to a constant number of the clusters, random sampling will tend to pick most of the centers from these relatively few clusters.

We also ran experiments on the following data sets taken from standard applications of k-means in vector quantization and pattern classification.

Lena22 and Lena44: These were taken from an application in image compression through vector quantization. The data were generated by partitioning a 512×512 gray-scale Lena image into 65,536 2×2 tiles. Each tile is treated as a point in a 4-dimensional space. Lena44 was generated using 4×4 tiles, thus generating 16,384 points in 16-dimensional space. We considered $k \in \{8, 64, 256\}$.

Kiss: This is from a color quantization application. 10,000 RGB pixel values were sampled at random from a color image of a painting "The Kiss" by Gustav Klimt. This resulted in 10,000 points in 3-space. We considered $k \in \{8, 64, 256\}$.

Forest: This data set came from the UCI Knowledge Discovery in Databases Archive. The data set relates forest cover type for 30×30 meter cells, obtained from the US Forest Service. The first 10 dimensions contain integer quantities, and the remaining 44 are binary values (mostly 0's). We sampled 10,000 points at random from the entire data set of 581,012 points in dimension 54. We considered $k \in \{10, 50, 100\}$.

For all heuristics the initial centers were taken to be a random sample of the point set. For the sake of consistency, for each run the various heuristics were started with the same set of initial centers. Each time the set of centers is changed, the distortion is recomputed. The combination of modifying the set of centers and recomputing distortions is called a *stage*. We measured convergence rates by tracking the lowest distortion encountered as a function of the number of stages executed. We also computed the average CPU time per stage. We use the filtering algorithm from [24] for computing distortions for all the heuristics. The results in each case were averaged over five trials having different random data points (for the synthetic examples) and different random initial centers. We ran the swap heuristic for $p \in \{1, 2\}$ swaps. Because they lack a consistent termination condition, all heuristics were run for 500 stages.

5.1. Comparison of convergence rates

In order to compare the quality of the clustering produced by the various heuristics, we ran each heuristic for 500 stages and plotted the best average distortion after each stage. These plots are shown in Fig. 6 for the ClusGauss, MultiClus and Lena44 data sets.

A number of observations can be made from these plots. After a small number of stages both iterated Lloyd's and the hybrid algorithms converged rapidly. However, after this initial start the iterated Lloyd's algorithm rarely makes significant gains in distortion. The problem is that this algorithm begins each run with an entirely new set of random centers, without accounting for which centers were well placed and which were not. In contrast, the swap heuristics tend to converge very slowly, and even after 500 stages they do not surpass the progress that the iterated Lloyd's algorithm makes in its first 10–50 stages. Since these heuristics do not use Lloyd's algorithm for descending to a local minimum, their gains occur only through the relatively slow process of making good random choices. As expected, the hybrid method does best of all. It has the same rapid initial convergence as with the iterated Lloyd's algorithm, but through repeated swaps, it can transition out of local minima. For most of the real data sets, the hybrid method and Lloyd's method produce very similar distortions. (This is not surprising, given the popularity of Lloyd's algorithm over many years.) Nonetheless, we observed instances where the hybrid method performs

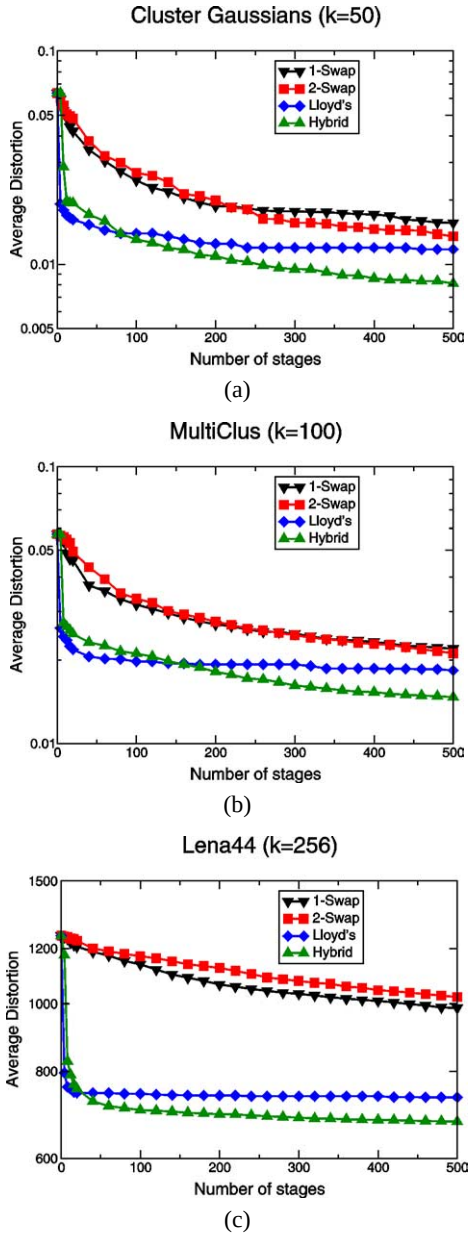

Fig. 6. Comparison of the average distortions versus number of stages of execution for ClusGauss ($k = 50$), MultiClus ($k = 100$), and Lena44 ($k = 256$). Note that the y-axis is plotted on a log scale and does not start from 0.

significantly better than the iterated Lloyd's algorithm, and we never found it to perform significantly worse. The hybrid algorithm tends to outperform the iterated Lloyd's algorithm in instances involving large numbers of well separated clusters.

Our results comparing the performance on all the data sets is given in Table 1. It shows the best distortions at stages 50, 100 and 500, and CPU times. To facilitate comparison, single-swap and single-

Table 1
Summary of experiments. Absolute values are indicated for Lloyd's algorithm, and the other values are given as a percentage of increase (positive) or decrease (negative) relative Lloyd's algorithm

DataSet	k	Method	Best distortion			Time/Stage
Size/Dim			Stage 50	Stage 100	Stage 500	(CPU sec)
		Lloyd's	0.048834	0.045096	0.041236	0.00989
	25	1-swap	80.2%	61.2%	41.3%	10.0%
		hybrid	2.3%	−0.2%	−7.6%	−24.8%
ClusGauss		Lloyd's	0.014546	0.013956	0.011758	0.01852
$n = 10{,}000$	50	1-swap	131.6%	92.3%	15.0%	−8.0%
$d = 3$		hybrid	15.7%	−6.4%	−30.9%	−18.7%
		Lloyd's	0.005953	0.005914	0.005868	0.03318
	100	1-swap	141.7%	104.2%	22.4%	−2.0%
		hybrid	6.1%	−0.6%	−2.9%	1.2%
		Lloyd's	0.036752	0.03633	0.03428	0.02437
	50	1-swap	83.6%	49.9%	11.1%	−15.1%
		hybrid	1.5%	−7.7%	−16.6%	−27.7%
MultiClus		Lloyd's	0.020258	0.01981	0.01839	0.03658
$n = 10{,}000$	100	1-swap	100.5%	68.5%	15.3%	−6.9%
$d = 3$		hybrid	12.7%	6.8%	−20.0%	−18.5%
		Lloyd's	0.004123	0.00393	0.00372	0.11064
	500	1-swap	194.0%	186.7%	102.7%	4.2%
		hybrid	4.2%	2.3%	−13.3%	−6.3%
		Lloyd's	349.28	342.48	339.62	0.07312
	8	1-swap	26.6%	21.7%	10.6%	1.7%
		hybrid	0.4%	0.2%	−0.3%	1.5%
Lena22		Lloyd's	107.82	107.00	106.32	0.29192
$n = 65{,}536$	64	1-swap	38.8%	32.2%	16.5%	−1.0%
$d = 4$		hybrid	−0.2%	−1.9%	−4.3%	−7.6%
		Lloyd's	56.35	56.35	55.54	0.57020
	256	1-swap	63.4%	55.9%	33.8%	4.9%
		hybrid	−3.3%	−5.8%	−7.8%	−8.5%
		Lloyd's	2739.2	2720.0	2713.2	0.20412
	8	1-swap	20.2%	11.4%	7.4%	4.6%
		hybrid	1.1%	0.7%	0.0%	1.2%
Lena44		Lloyd's	1158.8	1156.2	1150.4	1.19340
$n = 16{,}384$	64	1-swap	40.9%	34.8%	21.1%	−1.4%
$d = 16$		hybrid	−0.9%	−1.7%	−3.5%	−5.7%
		Lloyd's	744.7	742.7	734.2	−3.14580
	256	1-swap	60.2%	57.5%	39.3%	7.7%
		hybrid	−3.5%	−5.2%	−7.7%	20.3%

(*continued on next page*)

swap hybrid are given as percentage of increase relative to Lloyd's. (In particular, letting L and H denote the performance quantities for Lloyd's algorithm and another algorithm respectively, the listed percentage is $100(H - L)/L$.) The 2-swap heuristic performed very similarly to single-swap and is not shown here.

Table 1 (*Continued from previous page*)

DataSet Size/Dim	k	Method	Best distortion Stage 50	Best distortion Stage 100	Best distortion Stage 500	Time/Stage (CPU sec)
		Lloyd's	705.88	703.50	693.56	0.01062
	8	1-swap	34.9%	20.5%	9.2%	−2.4%
		hybrid	5.6%	0.8%	−0.4%	−2.5%
Kiss		Lloyd's	156.40	153.32	147.44	0.03528
$n = 10{,}000$	64	1-swap	86.6%	62.8%	20.7%	1.0%
$d = 3$		hybrid	1.9%	−1.4%	−7.0%	−6.2%
		Lloyd's	60.71	60.34	59.13	0.07621
	256	1-swap	85.2%	76.4%	34.3%	1.8%
		hybrid	−0.2%	−2.3%	−11.0%	−7.3%
		Lloyd's	595040	588860	587340	0.13722
	10	1-swap	28.9%	26.3%	19.5%	−1.4%
		hybrid	0.7%	0.8%	−0.7%	−14.6%
Forest		Lloyd's	202980	199360	198140	0.38842
$n = 10{,}000$	50	1-swap	56.6%	46.4%	26.0%	7.2%
$d = 54$		hybrid	−0.3%	−0.4%	−3.7%	−14.1%
		Lloyd's	138600	137800	136280	0.62256
	100	1-swap	62.6%	50.7%	28.0%	11.7%
		hybrid	−0.9%	−2.1%	−4.5%	−10.5%

Again, with respect to average distortions, the hybrid algorithm never performed significantly worse than the other heuristics, and sometimes performed significantly better. It is also interesting to observe that the hybrid method's running time is generally as good, if not better, than the other heuristics. Execution time will be discussed further in Section 5.2.

The fundamental question, which we cannot answer, is how good are these heuristics relative to the optimum. Because we do not know the optimal distortion, we can only compare one algorithm against another. In the case of the ClusGauss, however, it is possible to estimate the optimal distortion. In dimension 3, with $k = 50$ and $\sigma = 0.05$, the expected squared distance from each generated data point is $3\sigma^2 = 0.0075$. After 500 iterations, the hybrid method achieved an average distortion of 0.00813, which is about 8.4% above the expected optimal value (see Fig. 6(a)). The relatively good performance of the hybrid algorithm relative to the other heuristics suggests that, at least for the relatively sets that we tested, the hybrid heuristic's performance is much closer to optimal than our proven approximation bounds would suggest.

5.2. Parametric analysis of performance

In order to better understand the performance of the various heuristics as a function of the parameters involved, we ran a number of experiments in which we varied the sizes of the various quantities. All experiments involved the ClusGauss distribution, where the number of clusters was adjusted to match the number k of centers computed. The parameters we varied included the number n of data points, the number k of centers, the dimension d, and the coordinate standard deviation σ for the Gaussian clusters.

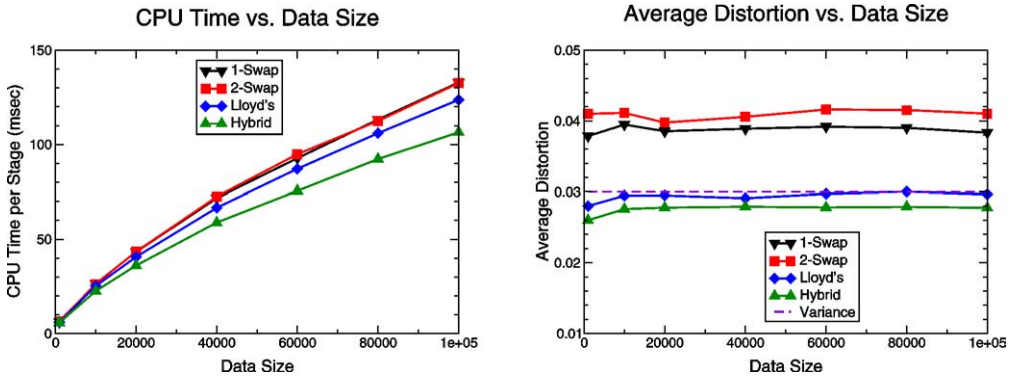

Fig. 7. CPU time and average distortion versus number of points. ($n = 10{,}000$, $k = 50$, $\sigma = 0.1$, $d = 3$.)

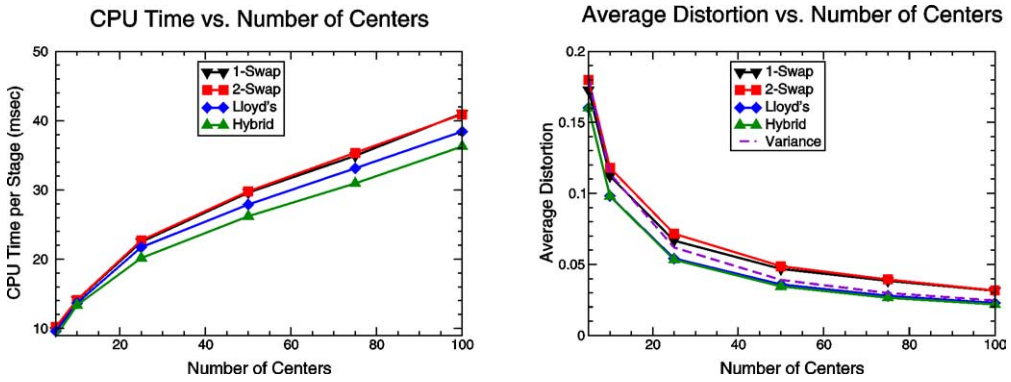

Fig. 8. CPU time and average distortion versus number of centers. ($n = 10{,}000$, $\sigma \approx k^{-1/3}/2$, $d = 3$.)

In each case we ran the heuristic for 500 iterations and recorded the running time in CPU seconds and the average distortion.

When varying the number k of centers or the dimension d, we also adjusted the value of σ, so that the clusters were similarly well separated. Recall that the cluster centers are uniformly distributed in a hypercube of side length 2. Intuitively, if we subdivide this hypercube into a grid of subcubes each of side length $(2/k)^{1/d}$, the expected number of clusters centers per subcube is exactly 1. Assuming an ideal situation in which each cluster center is located at the center of each subcube, this would imply an ideal separation distance of $(2/k)^{1/d}$ between neighboring subcubes. To model this, we generated clusters with a coordinate standard deviation of $c(2/k)^{1/d}$, for some constant $c < 1$. Of course, some clusters will be more well separated than others due to random variations in the placement of cluster centers, but we felt that this adjustment would help better distinguish variations due solely to k and d from variations due to cluster separation.

One advantage of having moderately well separated clusters is that we can use the cluster variance as a rough estimate for the optimal distortion. As clusters tend to overlap, the optimum distortion will tend to be lower, since outlying points generated from one Gaussian cluster may be assigned to a closer center. In our plots in Figs. 7–10 of average distortion, we have shown this variance-based distortion estimate as a broken line, to give a better sense of the optimum distortion.

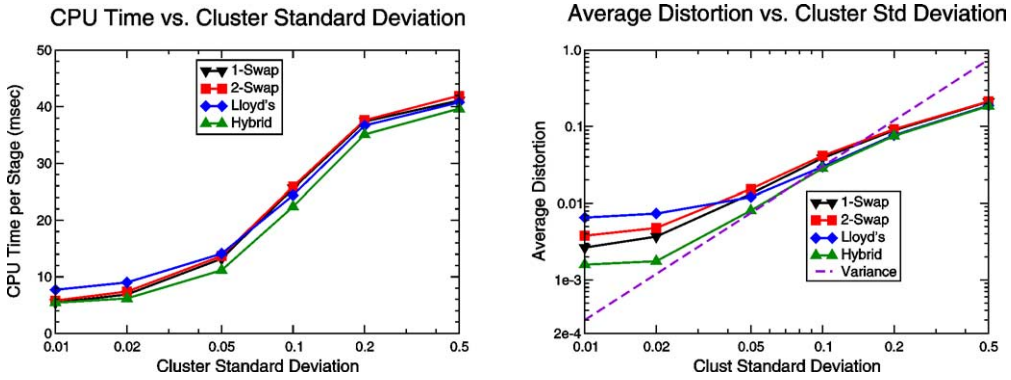

Fig. 9. CPU time and average distortion versus cluster standard deviation. ($n = 10{,}000$, $k = 50$, $d = 3$.)

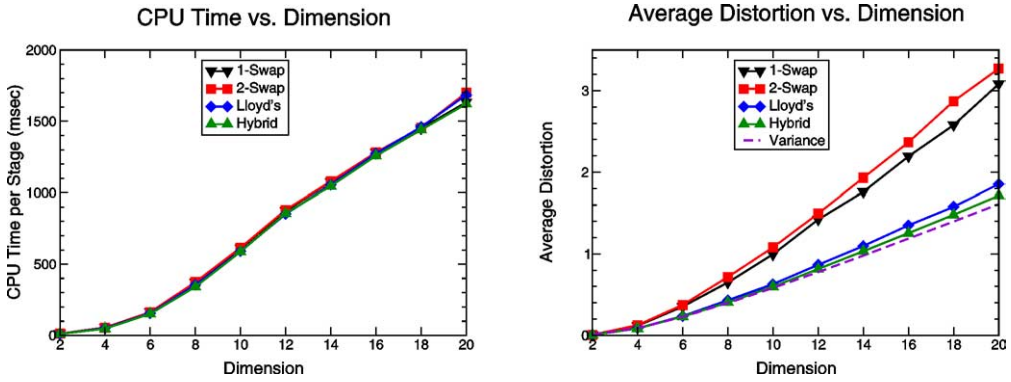

Fig. 10. CPU time and average distortion versus dimension. ($n = 10{,}000$, $k = 50$, $\sigma \approx k^{-1/d}/2$.)

As mentioned above, all the heuristics use the same filtering algorithm [24] for computing nearest centers and distortions. Since this is the dominant component of the running time, we observed that all the heuristics had very similar running times. The filtering algorithm uses a kd-tree to store the data points and uses a pruning technique to compute nearest centers and distortions. As a result, its running time is expected to be sublinear in n and k, assuming that the dimension d is fixed. See [24] for further analysis of this algorithm. (In contrast, a brute-force implementation of the nearest center computation would require $O(dkn)$ time.)

Varying data size: In this experiment, the number n of data points was varied from 1000 to 100,000, fixing $k = 50$, $d = 3$ and $\sigma = 0.1$. The results are shown in Fig. 7. As expected, the running times grow sublinearly with n. The hybrid heuristic and iterated Lloyd's achieved the best average distortions.

Varying the number of centers: Here the number k of centers was varied from 5 to 100, while fixing $n = 10{,}000$ and $d = 3$. We generated k Gaussian clusters in each case. As explained above, in order to guarantee similar cluster separation, we set the standard deviation $\sigma = (1/k)^{1/3}/3$ for each coordinate. The results are shown in Fig. 8. As expected, running times grow sublinearly

with k, and, as the number of centers grew, the average distortion decreased. All the heuristics produced similar average distortions.

Varying cluster standard deviation: Here we varied the standard deviation of the generated clusters from 0.01 (highly separated clusters) to 1 (overlapping clusters). We fixed $n = 10{,}000$, $k = 50$ and $d = 3$. The results are shown in Fig. 9. Running times were seen to increase as the clusters are less well separated. This effect is anticipated in the analysis of the filtering algorithm given in [24]. When the clusters are well separated, the hybrid heuristic tends to produce the smallest average distortions. In the absence of well defined clusters, all the heuristics produced similar distortions.

Varying dimension: The dimension was varied, while fixing $n = 10{,}000$ and $k = 50$. To maintain similar cluster separation, we set σ to $(1/k)^{1/d}/3$. The results are shown in Fig. 10. As with many algorithms based on hierarchical spatial subdivision, the running time of the filtering algorithm grows superlinearly with dimension. The curse of dimensionality would suggest that the growth rate should be exponential in dimension, but these experiments indicate a more modest growth. This is likely due to boundary effects. This phenomenon was described in [4] in the context of nearest neighbor searching. The hybrid heuristic and iterated Lloyd's performed comparably with respect to average distortion, while the swap heuristics performed considerably worse. This suggests that the importance of moving to a local minimum grows in significance as dimension increases.

6. Conclusions

We have presented an approximation algorithm for k-means clustering based on local search. The algorithm achieves a factor $9 + \varepsilon$ approximation ratio. We presented an example showing that any approach based on performing a fixed number of swaps achieves an approximation factor of at least $(9 - \varepsilon)$ in all sufficiently high dimensions. Thus, our approximation factor is almost tight for this class of local search algorithms. We have also presented empirical evidence that by combining this algorithm with Lloyd's algorithm (a simple descent algorithm, which produces a locally minimal solution) the resulting hybrid approach has very good practical performance.

This work provides further insights into k-means and other geometric clustering problems from both a practical and theoretical perspective. This work shows that it is possible to provide theoretical performance guarantees (albeit weak ones) on the performance of simple heuristics. It also shows the practical value of combining discrete approximation algorithms with continuous approaches that produce locally optimal solutions.

There are a number of open problems. Our analysis shows that if only single swaps are performed, the best approximation bound is $25 + \varepsilon$. However, we know of no centroidal configuration in any dimension for which the algorithm is at a stable configuration and the performance ratio is worse than $9 - \varepsilon$. Furthermore, in our tight example, we assume that the dimension may be chosen as a function of the number of swaps. This raises the question of whether a tighter analysis might show that an approximation factor better than 25 can be achieved even for single swaps and/or in fixed dimensions. Our analysis makes use of the fact that the optimal solution is centroidal. By alternating steps of the swap algorithm with Lloyd's algorithm, it is possible to assume that the heuristic solution is centroidal as well. Could

such an assumption be used to tighten our analysis? A final important question needed for empirical analysis of the approximation bounds is how to generate good lower bounds on the optimal distortion.

Acknowledgements

We would like to thank Hiroshi Imai and Mary Inaba for motivating our initial interest in k-means clustering. We would also like to thank David Kirkpatrick and Sergei Bespamyatnikh for interesting discussions on convergence properties of Lloyd's algorithm. We would also like to thank the referees for a number of suggestions, which improved the quality of the final paper.

References

[1] P.K. Agarwal, C.M. Procopiuc, Exact and approximation algorithms for clustering, in: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 658–667.
[2] K. Alsabti, S. Ranka, V. Singh, An efficient k-means clustering algorithm, in: Proceedings of the First Workshop on High Performance Data Mining, Orlando, FL, 1998.
[3] S. Arora, P. Raghavan, S. Rao, Approximation schemes for Euclidean k-median and related problems, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, pp. 106–113.
[4] S. Arya, D.M. Mount, O. Narayan, Accounting for boundary effects in nearest-neighbor searching, Discrete Comput. Geom. 16 (1996) 155–176.
[5] V. Arya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, K. Munagala, Local search heuristics for k-median and facility location problems, in: Proceedings of the 33rd Annual Symposium on Theory of Computing, Crete, Greece, 2001, pp. 21–29.
[6] G.H. Ball, D.J. Hall, Some fundamental concepts and synthesis procedures for pattern recognition preprocessors, in: International Conference on Microwaves, Circuit Theory, and Information Theory, Tokyo, Japan, 1964.
[7] S. Bandyopadhyay, U. Maulik, M.K. Pakhira, Clustering using simulated annealing with probabilistic redistribution, Internat. J. Patt. Recog. Artif. Intell. 15 (2001) 269–285.
[8] V. Capoyleas, G. Rote, G. Woeginger, Geometric clusterings, J. Algorithms 12 (1991) 341–356.
[9] M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k-medians problem, in: Proceedings of the 4th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 378–388.
[10] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tesselations: Applications and algorithms, SIAM Rev. 41 (1999) 637–676.
[11] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[12] A.E. ElGamal, L.A. Hemanchandra, I. Shperling, V.K. Wei, Using simulated annealing to design good codes, IEEE Trans. Inform. Theory 33 (1987) 116–123.
[13] D. Eppstein, Faster construction of planar two-centers, in: Proc. 8th ACM-SIAM Sympos. Discrete Algorithms, 1997.
[14] V. Faber, Clustering and the continuous k-means algorithm, Los Alamos Sci. 22 (1994) 138–144.
[15] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996.
[16] W. Feller, An Introduction to Probability Theory and its Applications, third edition, Wiley, New York, 1968.
[17] E. Forgey, Cluster analysis of multivariate data: Efficiency vs. interpretability of classification, Biometrics 21 (1965) 768.
[18] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[19] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic, Boston, MA, 1992.
[20] M. Inaba, N. Katoh, H. Imai, Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering, in: Proceedings of the Tenth Annual ACM Symposium on Computational Geometry, Stony Brook, NY, 1994, pp. 332–339.

[21] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice Hall, Englewood Cliffs, NJ, 1988.
[22] A.K. Jain, P.W. Duin, J. Mao, Statistical pattern recognition: A review, IEEE Trans. Patt. Anal. Mach. Intell. 22 (1) (2000) 4–37.
[23] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: A review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[24] T. Kanungo, D.M. Mount, N.S. Netanyahu, C. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. Patt. Anal. Mach. Intell. 24 (2002).
[25] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[26] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[27] T. Kohonen, Self-Organization and Associative Memory, third edition, Springer-Verlag, New York, 1989.
[28] S. Kolliopoulos, S. Rao, A nearly linear-time approximation scheme for the Euclidean k-median problem, in: J. Nesetril (Ed.), Proceedings of the Seventh Annual European Symposium on Algorithms, Lecture Notes Comput. Sci., vol. 1643, Springer-Verlag, Berlin, 1999, pp. 362–371.
[29] M. Korupolu, C. Plaxton, R. Rajaraman, Analysis of a local search heuristic for facility location problems, in: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 1–10.
[30] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (1982) 129–137.
[31] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Berkeley, CA, 1967, pp. 281–296.
[32] J. Matoušek, On approximate geometric k-clustering, Discrete Comput. Geom. 24 (2000) 61–84.
[33] R.R. Mettu, C.G. Plaxton, Optimal time bounds for approximate clustering, in: Proc. 18th Conf. on Uncertainty in Artif. Intell., Edmonton, Canada, 2002, pp. 339–348.
[34] R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in: Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, 1994, pp. 144–155.
[35] D. Pelleg, A. Moore, Accelerating exact k-means algorithms with geometric reasoning, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, 1999, pp. 277–281.
[36] D. Pelleg, A. Moore, x-means: Extending k-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, Palo Alto, CA, 2000.
[37] S.J. Phillips, Acceleration of k-means and related clustering problems, in: D.M. Mount, C. Stein (Eds.), Algorithm Engineering and Experiments (Proc. ALENEX '02), Lecture Notes Comput. Sci., vol. 2409, Springer-Verlag, Berlin, 2002.
[38] S.Z. Selim, M.A. Ismail, K-means-type algorithms: A generalized convergence theorem and characterization of local optimality, IEEE Trans. Patt. Anal. Mach. Intell. 6 (1984) 81–87.
[39] M. Sharir, A near-linear algorithm for the planar 2-center problem, Discrete Comput. Geom. 18 (1997) 125–134.
[40] M. Thorup, Quick k-median, k-center, and facility location for sparse graphs, in: Proc. 28th Intl. Colloq. on Automata, Languages and Programming (ICALP), Lecture Notes Comput. Sci., vol. 2076, Springer-Verlag, Berlin, 2001, pp. 249–260.
[41] J. Vaisey, A. Gersho, Simulated annealing and codebook design, in: Proc. IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 1988, pp. 1176–1179.
[42] G. Wesolowsky, The Weber problem: History and perspective, Location Sci. 1 (1993) 5–23.