

An Iterated Local Search Approach for Minimum Sum-Of-Squares Clustering

Peter Merz

University of Kaiserslautern
Department of Computer Science
Postbox 3049, D-67653 Kaiserslautern, Germany
`peter.merz@ieee.org`

Abstract. Since minimum sum-of-squares clustering (MSSC) is an *NP*-hard combinatorial optimization problem, applying techniques from global optimization appears to be promising for reliably clustering numerical data. In this paper, concepts of combinatorial heuristic optimization are considered for approaching the MSSC: An iterated local search (ILS) approach is proposed which is capable of finding (near-)optimum solutions very quickly. On gene expression data resulting from biological microarray experiments, it is shown that ILS outperforms multi-start *k*-means as well as three other clustering heuristics combined with *k*-means.

1 Introduction

Clustering can be considered as an optimization problem in which an assignment of data vectors to clusters is desired, such that the sum of squared distances of the vectors to their cluster mean (centroid) is minimal. Let \mathcal{P}_k denote the set of all partitions of X with $X = \{x_1, \dots, x_n\}$ denoting the data set of vectors with $x_i \in \mathbb{R}^m$ and C_i denoting the i -th cluster with mean \hat{x}_i . Thus, the objective is

$$\min_{\mathcal{P}_k \in \mathcal{P}_k} \sum_{i=1}^k \sum_{x_j \in C_i} d^2(x_j, \hat{x}_i), \quad \text{with} \quad \hat{x}_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \quad (1)$$

where $d(\cdot, \cdot)$ is the Euclidean distance in \mathbb{R}^m . Alternatively, we can formulate the problem as searching for an assignment p of the vectors to the clusters with $C_i = \{j \in \{1, \dots, n\} \mid p[j] = i\}$. Thus, the objective becomes

$$\min_p \sum_{i=1}^k d^2(x_i, \hat{x}_{p[i]}). \quad (2)$$

This combinatorial problem is called the minimum sum-of-squares clustering (MSSC) problem and is known to be *NP*-hard [1]. Although exact optimization algorithms for clustering criteria exist [2, 3], their application is limited to small data sets.

Clustering algorithms have received renewed attention in the field of bioinformatics due to the breakthrough of microarrays. This technology allows monitoring simultaneously the expression patterns of thousands of genes with enormous promise to help genetics to understand the genome [4, 5]. Since a huge amount of data is produced during microarray experiments, clustering techniques are used to extract the fundamental patterns of gene expression inherent in the data. Several clustering algorithms have been proposed for gene expression profile analysis: Eisen *et al.* [6] applied a hierarchical average link clustering algorithm to identify groups of co-regulated genes in the yeast cell cycle. In [7], the k -means algorithm was used for clustering yeast data and in [8], a comparison of several approaches was made including k -means with average link initialization, which performed well for all tested data sets.

The k -means algorithm is a heuristic which minimizes the sum-of-squares criterion provided an initial assignment/choice of centroids. The number k of clusters is fixed during its run. The k -means heuristic can in fact be regarded as a local search heuristic for this hard combinatorial optimization problem. It is well-known that the effectiveness of k -means highly depends on the initial choice of centroids [9, 10].

Since iterated local search (ILS) is known to be highly effective for several combinatorial optimization problems such as the traveling salesman problem [11], the application of ILS to the MSSC appears to be promising.

In this paper, iterated local search for clustering gene expression profiles using the k -means heuristic is proposed. It is shown that instead of repeatedly starting k -means to find better local optima, the ILS framework is much more effective in finding near optimum solutions quickly for the investigated data sets.

The paper is organized as follows. The general ILS framework is described in section 2. In section 3, an ILS approach to minimum-sum-of-squares clustering is introduced. In section 4, distance measures for comparing clustering solutions are provided. The ILS approach is applied for clustering gene expression data in section 5. Section 6 concludes the paper and outlines areas for future research.

2 Iterated Local Search

Per definition, local search heuristics suffer from the fact that they get trapped in local optima with respect to some objective function. In fact, for NP -hard combinatorial problems, there is no known polynomial time algorithm that guarantees to find the optimum solution. Hence, the only practical way is to approach these problems with heuristics. Local search algorithms are improvement heuristics which often produce near-optimum solutions very quickly. To improve these local optimum solutions, additional techniques are required. Iterated Local Search (ILS) [12] is such a technique which is very simple but known to be powerful. A famous example is the iterated Lin-Kernighan heuristic for the traveling salesman problem [11], which is one of the best performing heuristics for the traveling salesman problem to date.

The basic idea behind ILS is as follows: A local search is applied to an initial solution either generated randomly or with some constructive heuristic. Then, repeatedly, the current local optimum is mutated, i. e. slightly modified, and the local search procedure is applied to reach another local optimum. If the new local optimum has a better objective value, the solution is accepted as the new current solution. The outline of the algorithm is provided in Fig. 1.

```

procedure Iterated-Local-Search( $n_{max} : N$ ) :  $S$ ;
begin
  Create starting solution  $s$ ;
   $s :=$  Local-Search( $s$ );
   $n := 0$ ;
  repeat
     $s' :=$  Mutate( $s$ );
     $s' :=$  Local-Search( $s'$ );
    if  $f(s') < f(s)$  then  $s := s'$ ;
     $n := n + 1$ ;
  until  $n = n_{max}$ ;
  return  $s$ ;
end;

```

Fig. 1. Iterated Local Search pseudo code.

ILS provides a simple extension to local search. In contrast to Multi-Start Local search (MLS), in which repeatedly local search is applied to newly randomly generated solutions, some information is preserved from previous iterations in ILS.

ILS can be considered as an evolutionary algorithm incorporating local search. In fact, it is a special case of a memetic algorithm [13, 14], with a population size of one. In terms of evolutionary algorithms ILS can be regarded as a (1+1)-ES (Evolution Strategy) with additional local search. Compared to common evolutionary algorithms, mutation is aimed at escaping from a local optimum and thus much more disruptive than mutation for neighborhood searching.

3 ILS for Clustering

There are two operators in ILS specific to the problem: (a) the local search, and (b) the mutation operator. Both operators used in our ILS for MSSC are described in the following.

3.1 Initialization and Local Search

In our ILS, local search is performed using the k -means heuristic [15, 16]: Given k centroid vectors a_j , for each input vector $x_i \in X$ the nearest centroid s_j is determined and the vector x_i is associated with the corresponding cluster. Afterwards, all centroid vectors are recalculated by calculating the mean of the vectors in each cluster. This procedure is repeated until the partition does no longer change, and thus a local optimum is reached.

The initial solutions in ILS for MSSC are generated by randomly selecting input vectors as initial cluster centers for the k -means.

Variants to the k -means heuristic above may be used within the framework, such as k -means using k -d trees [17–19] and X -means [20]. For prove of concept we concentrate on the simple k -means described above.

3.2 The Mutation Operator

The mutation operator used in ILS works as follows: A randomly chosen mean vector a_j ($1 \leq j \leq K$) is replaced by a randomly chosen input vector x_i ($1 \leq i \leq N$).

4 Comparing Clustering Solutions

To allow a comparison of clusters produced by the clustering algorithms used in our studies, we define two distance measures between clustering solutions. The first defines a distance between two assignments p and q by the formula

$$D(p, q) = \sum_{i=1}^n d^2(\hat{x}_{p[i]}, \hat{x}_{q[i]}). \quad (3)$$

We will refer to this distance measure as the *means distance*. However, the value of this distance metric cannot be interpreted easily. It would be helpful to have a measure indicating how many input vectors have been assigned to different clusters. Furthermore, a measure independent of the values of the centroids allows to compare clustering solutions obtained by other heuristics not using centroids at all.

Therefore, we propose another technique called *matching* to provide a measure for (dis-)similarity of clustering solutions (assignments). Matching works as follows. First, a counter for common vectors is set to zero. Then, for each cluster i of solution A , find the cluster j of solution B containing the most vectors of cluster i . Additionally, find the cluster k from solution A containing most vectors of cluster j . If k equals i , we have a matching and the counter is incremented by the number of vectors contained in cluster i of solution A and also in cluster j of solution B . If we have done the matching for all clusters in solution A , the counter contains the total number of vectors assigned to matching clusters. The distance of solutions A and B is simply the difference of the total number

of input vectors and the number of matching vectors. This distance measure is easily interpreted since it provides a measure of how many vectors have been assigned to 'different' clusters in the two solutions. In the following, we refer to this distance as the *matching distance*.

5 Computational Experiments

We compared the iterated local search (ILS) utilizing the mutation operator described above with a multi-start k -means local search (MLS). In the MLS, a predefined number of times a start configuration is randomly generated and the k -means algorithm is applied. The best solution found is reported.

All algorithms were implemented in Java 1.4. Instead of comparing absolute running times, we provide the number of k -means iterations performed by the algorithms to have a measure independent of programming language or code optimizations.

5.1 The Gene Expression Data Sets

The first data set denoted as HL-60 is taken from [21] and contains data from macrophage differentiation experiments. The data consists of 7229 genes and expression levels at 4 time points. We applied a variation filter which discarded all genes with an absolute change in expression level less than or equal to 5. The number of genes which passed the filter was 3230. The vectors were normalized afterwards to have mean 0 and variance 1, as described in [7].

The second data set denoted as HD-4CL is also taken from [21] and contains data from hematopoietic differentiation experiments across 4 cell lines. The data consists of 7229 genes, 17 samples each. We applied the same variation filter as above which discarded all genes with an absolute change in expression level less than or equal to 5. The number of genes which passed the filter was 2046. Afterwards, the vectors were normalized to have mean 0 and variance 1.

The third data set is denoted as Cho-Yeast and is described in [22]. It contains the expression levels of 6565 yeast genes measured at 17 time points over two complete cell cycles. As in [7], we discarded the time points at 90 and 100 min, leading to a 15 dimensional space. A variation filter was used which discarded all genes with an absolute change in expression level less than or equal to 50 and an expression level of $\max/\min < 2.0$. The resulting number of genes was 2931. Again, the vectors were normalized afterwards to have mean 0 and variance 1.

To study the capability of ILS to find globally optimum solutions, the fourth and fifth data set were randomly generated with 5000 and 6000 vectors, denoted DS-5000 and DS-6000, respectively. The main vectors were generated with 16 time points: For each cluster, the cluster center $x = (x_1, \dots, x_{16})$ was generated as follows. x_1 was chosen with a uniform random distribution, and all other x_i by an AR(1) process such that $x_{i+1} = x_i + \epsilon_i$, where ϵ_i is a normally distributed random number. All other cluster members were chosen to be normally

distributed 'around' the cluster center. No variation filter was applied and the vectors were normalized as described above.

The number of clusters for the clustering were taken from [7] for **Cho-Yeast** ($k = 30$), and from [21] for the data sets **HL-60** ($k = 12$), **HD-4CL** ($k = 24$). For the data sets **DS-5000** and **DS-6000**, k was set to the known number of clusters in the experiments: $k = 25$ and $k = 30$, respectively.

5.2 ILS Performance

In the experiments, the ILS algorithm was run with a maximum number of iterations n_{max} set to 2000. Analogously, the MLS was run by calling k -means 2000 times. This way, the number of local searches was set to 2000 in both cases.

The results for MLS and ILS are reported in Table 1. For each data set and

Table 1. Comparison of Iterated Local Search and Multi-Start Local Search

Data Set	Algorithm	No. LS	Iter LS	Best	Avg. Obj.	Excess
HL-60	MLS	2000.0	95896.7	1749.86	1749.90	0.00%
	ILS	164.9	5660.2	1749.85	1749.89	0.00%
HD4CL	MLS	2000.0	61435.9	12476.92	12493.51	0.47%
	ILS	1721.5	26186.2	12434.91	12441.91	0.05%
Cho-Yeast	MLS	2000.0	74632.5	16966.78	16984.32	0.46%
	ILS	2000.0	32652.8	16905.22	16915.77	0.08%
DS-5000	MLS	2000.0	35954.7	3192.35	3497.14	9.55%
	ILS	207.4	2560.4	3192.35	3192.35	0.00%
DS-6000	MLS	2000.0	40929.9	5868.59	6314.74	8.65%
	ILS	272.1	3474.9	5811.82	5811.82	0.00%

algorithm, the average number of local searches (No. LS), the average number of local search iterations (Iter LS), the best objective found (best), the average best objective found (Avg. Obj.), and the average percentage excess over the optimum or best-known solution (Excess) is displayed. For each algorithm, 30 independent runs were performed.

The results show that ILS is clearly superior to MLS. ILS achieves better average and best objective values than MLS for all tested data sets. Moreover, ILS is at least more than two times faster. As can be seen for the data set **Cho-Yeast**, the number of iterations within k -means is more than two times lower in ILS than in MLS. This is due to the fact that mutation changes a local optimum solution only slightly and thus a new local optimum is reached with much fewer k -means iterations. In case of the largest data sets **DS-5000** and **DS-6000**, the optimum partition is found in all runs with an average number of 207.4 and 372.1 ILS iterations, respectively.

The results of ILS are comparable with those reported in [23]. However, ILS outperforms the memetic algorithms in [23] on the data set **Cho-Yeast**, and the running times in terms of k -means iterations is lower for data sets **DS-5000** and **DS-6000**.

5.3 Comparison with other Heuristics

To assess the performance of our ILS approach, we conducted experiments with three other clustering algorithms from the bioinformatics literature. These algorithms use k -means but with a different initialization. The first algorithm denoted 'Average-Link+ k -means' is an agglomerative hierarchical clustering algorithm combined with k -means. Starting with each vector in its own cluster, it works by stepwise merging the two clusters with the greatest similarity. The cluster similarity is determined by the average pairwise similarity between the genes in the clusters. The algorithm terminates if a predefined number of clusters is produced and k -means has been applied to the resulting solution. This combination has been shown to be superior to several other clustering approaches [8] for gene expression profiles.

Table 2. Comparison of the solution quality of three heuristics for the MSSC

Data Set	Algorithm	Objective	Excess
HL-60	Average-Link + k -Means	1753.93	0.23 %
	Farthest-Split + k -Means	1818.74	3.94 %
	MST-Cut + k -Means	1756.33	0.37 %
HD-4CL	Average-Link + k -Means	12824.10	3.13 %
	Farthest-Split + k -Means	12881.83	3.59 %
	MST-Cut + k -Means	12666.80	1.86 %
Cho-Yeast	Average-Link + k -Means	17118.25	1.28 %
	Farthest-Split + k -Means	17196.08	1.74 %
	MST-Cut + k -Means	17313.77	2.43 %
DS-5000	Average-Link + k -Means	3865.94	21.1 %
	Farthest-Split + k -Means	4354.12	36.39 %
	MST-Cut + k -Means	4359.50	36.56 %
DS-6000	Average-Link k -Means	6954.48	19.66 %
	Farthest-Split + k -Means	9600.66	65.19 %
	MST-Cut + k -Means	8213.83	41.33 %

The second algorithm denoted 'Farthest-Split + k -Means' is a divisive hierarchical algorithm used in [7] to cluster the yeast data and works as follows. The first cluster center is chosen as the centroid of the entire data set and subsequent

centers are chosen by finding the data point farthest from the centers already chosen. If the desired number of clusters is reached, the process is terminated and the k -means heuristic is applied. The third algorithm denoted by 'MST-Cut + k -Means' is a combination of the k -Means heuristic and a heuristic utilizing minimum spanning trees as proposed in [24] for gene expression data clustering. The latter works by calculating the minimum spanning tree for the graph defined by the data set and cutting the tree at its $k - 1$ longest edges. The resulting connected components of the tree define the k clusters.

The three algorithms were applied to the same data sets as ILS above. The results are displayed in Table 2. In the table, the objective value (Objective) in terms of the minimum sum-of-squares criterion is displayed as well as the percentage excess over the optimum or best-known objective value (Excess) found by ILS is presented. As the results show, these sophisticated heuristics are not capable of finding the globally optimum solution. The first heuristic 'Average-Link + k -Means' performs reasonably well in all the cases and is only outperformed by 'MST-Cut + k -Means' on the data set HD-4CL. However, all heuristics can not achieve the clustering quality of ILS. In case of the largest instances, all heuristics are even outperformed by the multi-start k -means heuristic. From the combinatorial optimization perspective, these results are not surprising. In many cases, ILS or the more sophisticated memetic algorithms are known to outperform combinations of constructive heuristics and local search [11, 25, 26].

Since the cluster memberships are more important than the clustering error, the distances to the best-known solutions for the heuristics are shown in Table 3. Both matching distance and means distance are provided. Interestingly, for the

Table 3. Distances to best-known solutions found by ILS

Data Set	Algorithm	Matching Distance	Means Distance
Cho-Yeast	Average-Link + k -Means	852.0	5379.0
	Farthest-Split + k -Means	922.0	5744.0
	MST-Cut + k -Means	1012.0	6202.0
DS-5000	Average-Link + k -Means	3.0	247.0
	Farthest-Split + k -Means	155.0	707.0
	MST-Cut + k -Means	70.0	711.0
DS-6000	Average-Link + k -Means	4.0	656.0
	Farthest-Split + k -Means	428.0	2678.0
	MST-Cut + k -Means	33.0	1314.0

data sets DS-5000 and DS-6000 the heuristics produce solutions which are similar to the optimum solution in terms of cluster memberships. In case of the real data set Cho-Yeast, however, they appear to be far away from the best-known solution in terms of cluster memberships. Hence, to arrive at optimum clustering

solutions, it is not sufficient to use these heuristics proposed previously in the bioinformatics literature. ILS appears to be much more robust in finding (near)-optimum solutions to the *NP*-hard clustering problem. The results by ILS are similar to the results found by memetic algorithms [23], but the algorithmic framework is much simpler. Hence, at least for the data sets tested, the use of sophisticated evolutionary algorithms is not required.

6 Conclusions

An new approach for minimum sum-of-squares clustering has been proposed which is based on concepts of combinatorial heuristic optimization. The approach, called iterated local search (ILS), combines *k*-means with a simple and powerful mechanism to escape out of local optima. It has been shown on gene expression data from the bioinformatics literature, that ILS is capable of finding optimum clustering solutions for data sets up to 6000 genes. Compared to multi-start *k*-means, ILS is superior in respect to the best solution found and the average best solution found while requiring less than half of the computation time. Moreover, it is demonstrated that more sophisticated initialization mechanisms for *k*-means used in the bioinformatics literature such as average linkage clustering, clustering by splitting off the farthest data point, and clustering by partitioning minimum spanning trees does not lead to a clustering quality achieved by ILS. Compared to the recently proposed memetic algorithms for clustering, ILS performs comparable but requires much less implementation efforts.

There are several issues for future research. The estimation of the number of clusters has to be incorporated in the ILS algorithm. Moreover, the application of large data sets with more than 100000 data points should be addressed in further studies. Finally, different local search procedures and *k*-means variants should be tested within the ILS framework.

References

1. Brucker, P.: On the Complexity of Clustering Problems. *Lecture Notes in Economics and Mathematical Systems* **157** (1978) 45–54
2. Grötschel, M., Wakabayashi, Y.: A Cutting Plane Algorithm for a Clustering Problem. *Mathematical Programming* **45** (1989) 59–96
3. Hansen, P., Jaumard, B.: Cluster Analysis and Mathematical Programming. *Mathematical Programming* **79** (1997) 191–215
4. Zhang, M.: Large-scale Gene Expression Data Analysis: A New Challenge to Computational Biologists. *Genome Research* **9** (1999) 681–688
5. Brazma, A., Vilo, J.: Gene Expression Data Analysis. *FEBS Letters* **480** (2000) 17–24
6. Eisen, M., Spellman, P., Botstein, D., Brown, P.: Cluster Analysis and Display of Genome-wide Expression Patterns. In: *Proceedings of the National Academy of Sciences, USA*. Volume 95. (1998) 14863–14867

7. Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., Church, G.M.: Systematic Determination of Genetic Network Architecture. *Nature Genetics* **22** (1999) 281–285
8. Yeung, K., Haynor, D., Ruzzo, W.: Validating Clustering for Gene Expression Data. *Bioinformatics* **17** (2001) 309–318
9. Bradley, P.S., Fayyad, U.M.: Refining Initial Points for k-Means Clustering. In: Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (1998) 91–99
10. Penã, J.M., Lozano, J.A., Larranãga, P.: An Empirical Comparison of Four Initialization Methods for the k-Means Algorithm. *Pattern Recognition Letters* **20** (1999) 1027–1040
11. Johnson, D.S., McGeoch, L.A.: The Traveling Salesman Problem: A Case Study. In Aarts, E.H.L., Lenstra, J.K., eds.: *Local Search in Combinatorial Optimization*. Wiley and Sons, New York (1997) 215–310
12. Lourenco, H.R., Martin, O., Stützle, T.: Iterated Local Search. In Glover, F., Kochenberger, G., eds.: *Handbook of Metaheuristics*. Kluwer Academic Publishers (2003)
13. Moscato, P.: Memetic Algorithms: A Short Introduction. In Corne, D., Dorigo, M., Glover, F., eds.: *New Ideas in Optimization*. McGraw–Hill, London (1999) 219–234
14. Merz, P., Freisleben, B.: Memetic Algorithms for the Traveling Salesman Problem. *Complex Systems* **13** (2001) 297–345
15. Forgy, E.W.: Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications. *Biometrics* **21** (1965) 768–769
16. MacQueen, J.: Some Methods of Classification and Analysis of Multivariate Observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. (1967) 281–297
17. Alsabti, K., Ranka, S., Singh, V.: An Efficient Space-Partitioning Based Algorithm for the k-Means Clustering. In Zhong, N., Zhou, L., eds.: *Proceedings of the 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD-99)*. Volume 1574 of *Lecture Notes in Artificial Intelligence*., Berlin, Springer (1999) 355–359
18. Pelleg, D., Moore, A.: Accelerating Exact k-Means Algorithms with Geometric Reasoning. In Chaudhuri, S., Madigan, D., eds.: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, ACM Press (1999) 277–281
19. Likas, A., Vlassis, N., Verbeek, J.J.: The Global k-Means Clustering Algorithm. *Pattern Recognition* (**36**)
20. Pelleg, D., Moore, A.: X-means: Extending *K*-means with Efficient Estimation of the Number of Clusters. In: Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2000) 727–734
21. Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S., Golub, T.R.: Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Application to Hematopoietic Differentiation. In: *Proceedings of the National Academy of Sciences, USA*. Volume 96. (1999) 2907–2912
22. Cho, R.J., Campbell, M.J., Winzeler, E.A., Conway, S., Wodicka, L., Wolfsberg, T.G., Gabrielian, A.E., Landsman, D., Lockhart, D.J., Davis, R.W.: A Genome-wide Transcriptional Analysis of the Mitotic Cell Cycle. *Molecular Cell* **2** (1998) 65–73

23. Merz, P., Zell, A.: Clustering Gene Expression Profiles with Memetic Algorithms. In et al., J.J.M.G., ed.: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, PPSN VII. Lecture Notes in Computer Science 2439, Springer, Berlin, Heidelberg (2002) 811–820.
24. Xu, Y., Olman, V., Xu, D.: Clustering Gene Expression Data using a Graph-Theoretic Approach: An Application of Minimum Spanning Trees. *Bioinformatics* **18** (2002) 536–545
25. Merz, P., Freisleben, B.: Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-Partitioning. *Evolutionary Computation* **8** (2000) 61–91
26. Merz, P., Katayama, K.: Memetic Algorithms for the Unconstrained Binary Quadratic Programming Problem. *Bio Systems* (2002) To appear.