Enhanced Genetic Algorithm with K-Means for the Clustering Problem

N. Bouhmala, A. Viken, and J. B. Lønnum

***Abstract*—In this paper, an algorithm for the clustering problem using a combination of the genetic algorithm with the popular K-Means greedy algorithm is proposed. The main idea of this algorithm is to use the genetic search approach to generate new clusters using the famous two-point crossover and then apply the K-Means technique to further improve the quality of the formed clusters in order to speed up the search process.**

Experimental results demonstrate that the proposed genetic algorithm combined with K-Means converges faster while producing the same quality of the clustering compared to the standard genetic algorithm.

***Index Terms*—Clustering problem, genetic algorithm, K-means.**

I. Introduction

Cluster analysis has been a hot topic of research due to its applicability in many disciplines including market segmentation [1], image processing [2], web mining [3], and bio-informatics [4], [5] to name just a few. The Clustering Problemcan be defined as a tuple(X, A, R, Φ) where:

- X is finite set of n data objects: $X = \{O_1, O_2, ...O_n\}$,
- A is a finite set of attributes or features: $A = \{A_{O1}, A_{O2}, ... A_{On}\}$. Thus each data object $Oi \in X$ has a corresponding discrete set of attributes A_{Oi} from which it can beidentified,
- R is a relation defining the constraints on the resulting clusters. For any two clusters C_i and C_j, $C_i \cap C_j = \emptyset$, $\forall_i \neq j$, and $i, j \in 1, 2, ...k$. The relation to be respected by all the formed clusters is that no pairs ofclusters should have a data object in common,
- And Φ is the objective function used for evaluating the clustering.

A solution to the clustering problem requires the partitioning of the n data objects into a set of k clusters C_k such that Φ is optimized. The number of possible partitions for n objects into k clusters is given by is a Stirling number of the second kind, which is given by:

$$S_n^k = \frac{1}{k!}\sum_{i=0}^{i=k}(-1)^{k-i}\binom{k}{i} i^n$$

This shows the combinatorial explosion due to the number of possible solutions. Clearly, searching all possible clustering alternatives would not be feasible. The problem becomes much harder when the number of clusters is not known. In this case, then the number of different partitions is the sum of the Stirling numbers of the second kind:

$$\sum_{i=1}^{i=k_{max}} S_n^i$$

where k_{max} is the maximum number of clusters with $k_{max} \leq n$. Because of these two reasons, there is a considerable interest in the design of heuristics to solve the clustering problems using a cost function that quantifies the goodness of the clusters on the basis of the similarity or dissimilarity measures of the data objects. A commonly used cost function is the sum of squared distances of the data objects to their cluster representatives. Given a partition P and the cluster representatives C, the clustering is usually assessed by Eq. (1)

$$\Phi = \sum_{i=1}^{i=n} Dist\left(O_i, c_{p_i}\right) \tag{1}$$

where Dist is a distance function taken between a data object and the cluster center. Euclidean Distance is the most widely used distance function in the clustering context, and it is calculated using Eq. (2)

$$ED = Dist(O_1, O_2) = \sqrt{\sum_{i=1}^{k}\left(O_1^i - O_2^i\right)^2}. \tag{2}$$

The hardness of the problem has stimulated the search for efficient clustering approximation algorithms which can be broadly be divided into three main types: hierarchical, partitional, and local search methods. Hierarchical clustering algorithms [6]-[8] construct a hierarchy of clusters using either agglomerative or divisive style. The agglomerative style starts with each data object in its own cluster, and at each step, the closest pair of clusters are merged using a metric of cluster proximity. Different agglomerative algorithms differ in how the clusters arethe other hand, Non-hierarchical or partitional clustering [9]-[11] are based on iterative relocation of data objects between clusters. The set of data objects is divided into non-overlapping clusters such that each data object lies in exactly one cluster. The quality of the solution is measured by a clustering criterion. At each iteration, the algorithm improves the value of the criterion function until convergence is reached. The algorithms belonging to this class generate solutions from scratch by adding to an initially empty partial solution

Manuscript received October 13, 2014; revised January 25, 2015.

The authors are with Buskerud and Vestfold University College, Faculty of Technology, Norway (e-mail: noureddine.bouhmala@hbv.no).

DOI: 10.7763/IJMO.2015.V5.452

components, until a solution is complete. They are regarded as the fastest approximate methods, yet they often return solutions of inferior quality. Finally, local search methods constitute an alternative to the traditional partitional techniques. These techniques offer the advantage of being flexible. They can be applied to any problem (discrete or continuous) whenever there is a possibility for encoding a candidate solution to the problem, and a mean of computing the quality of any candidate solution through the so-called cost function. They have the advantage that they could escape more efficiently from local minima. They start from some initial solution and iteratively try to replace the current solution by a better one in the light of the cost function in an appropriately defined neighborhood of the current solution. Their performances depend highly on finding a tactical interplay between diversification and intensification. The former refers to the ability to explore many different regions of the search space, whereas the latter refers to the ability to obtain high quality solutions within those regions. Examples include genetic algorithms [12]-[14], Tabu Search [15], Grasp [16].

II. The Algorithm

A. K-Means Algorithm

The K-Means [17] is a simple and well known algorithm used for solving the clustering problem. The goal of the algorithm is to find the best partitioning of n objects into k clusters, so that the total distance between the cluster's members and its corresponding centroid, representative of the cluster is minimized. The algorithm uses an iterative refinement strategy using the following steps:

Steps 1. This step determines the starting cluster's centroids. A very common used strategy is to assign random *k*, different objects as being the centroids.

Steps 2. Assign each object to the cluster that has the closest centroid. In order to find the cluster with the most similar centroid, the algorithm must calculate the distance between all the objects and each centroid.

Steps 3. Recalculate the values of the centroids. The values of the centroid are updated by taking as the average of the values of the object's attributes that are part of the cluster.

Steps 4. Repeat Steps 2 and Step 3 iteratively until objects can no longer change clusters.

B. Genetic Algorithms

Genetic Algorithms [18] are stochastic methods for global search and optimization and belong to the group of Evolutionary Algorithms. They simultaneously examine and manipulate a set of possible solution. Given a specific problem to solve, the input to GAs is an initial population of solutions called individuals or chromosomes. A gene is part of a chromosome, which is the smallest unit of genetic information. Every gene is able to assume different values called allele. All genes of an organism form a genome which affects the appearance of an organism called phenotype. The chromosomes areencoded using a chosen representation and each can be thought of as a point in the search space of candidate solutions. Each individual is assigned a score (fitness) value that allows assessing its quality. The members of the initial population may be randomly generated or by using sophisticated mechanisms by means of which an initial population of high quality chromosomes is produced. The reproduction operator selects (randomly or based on the individual's fitness) chromosomes from the population to be parents and enters them in a mating pool. Parent individuals are drawn from the mating pool and combined so that information is exchanged and passed to off-springs depending on the probability of the cross-over operator. The new population is then subjected to mutation and enters into an intermediate population. The mutation operator acts as an element of diversity into the population and is generally applied with a low probability to avoid disrupting cross-over results. Finally, a selection scheme is used to up- date the population giving rise to a new generation. The individuals from the set of solutions which is called population will evolve from generation to generation by repeated applications of an evaluation procedure that is based on genetic operators. Over many generations, the population becomes increasingly uniform until it ultimately converges to optimal or near-optimal solutions. The next section explains in detail the genetic algorithm used for the clustering problem. Algorithm 1 shows the various steps used in the proposed genetic algorithm.

Algorithm 1: Enhanced genetic algorithm with K-means.

input : Problem P_0
output: Solution $C_{final}(P_0)$
1) begin
2) Generate initial population;
3) Evaluate the fitness of each individual in the population;
4) while (Not Convergence reached) do
5) Select individuals according to a scheme to reproduce;
6) Breed each selected pairs ofindividuals through crossover;
7) Apply K-Means if necessary to each offspring according to *Pk*−Means;
8) Evaluate the fitness of the intermediate population;
9) Replace the parent population witha new generation ;
10) end
11) end

C. Implementation Details

1) Fitness function

The Notion of fitness is fundamental to the application of genetic algorithms. It is a numerical value that expresses the performance of an individual (solution) so that different individuals can be compared. The fitness function used by our genetic algorithm is simply Eq. (1).

2) Representation

A representation is a mapping from thestate space of possible solutions to a state of encoded solutions within a particular datastructure. The encoding scheme used in this work is based on integer encoding. An individual or chromosome is represented using a vector of N positions, where N is the set of data objects. Each position corresponds to a particular data object, i.e, the ith position (gene)

represents the ith data object. Each gene has a value over the set{1,2,....k}. These values define the set of cluster labels.

3) Initial population

The initial population consists of individuals generated randomly in which each gene's allele is assigned randomly a label from the set of cluster labels.

4) Cross-over

The task of the cross-over operator is to reach regions of the search space with higher average quality. New solutions are created by combining pairs of individuals in the population and then applying a crossover operator to each chosen pair. The individuals are visited in random order. An unmatched individual i_l is matched randomly with an unmatched individual i_m. Thereafter, the two-point crossover opera- tor is applied using a cross-over probability to each matched pair of individuals. The two-point crossover selects two randomly points within a chromosome and then interchanges the two parent chromosomes be- tween these points to generate two new off- spring. Recombination can be defined as a process in which a set of configurations (solutions referred as parents) undergoes a transformation to create a set of configurations (referred as off-springs). The creation of these descendants involves the location and combinations of features extracted from the parents. The reason behind choosing the two point crossover is the results presented in [19] where the difference between the different crossovers are not significant when the problem to be solved is hard. In addition, the work conducted in[20] shows that the two-point crossover is more effective when the problem at hand is difficult to solve.

5) K-Means

By introducing local search at this stage, the search within promising areas is intensified. This local search should be able to quickly improve the quality of a solution produced by the crossover operator, with- out diversifying it into other areas of the search space. In the context of optimization, this rises a number of questions regarding how best to take advantage of both aspects of the whole algorithm. With regard to local search there are issues of which individuals will undergo local improvement and to what degree of intensity. However care should be made in order to balance the evolution component (exploration) against exploitation (local search component). Bearing this thought in mind, the strategy adopted in this regard is to let each chromosome go through a low rate intensity local improvement. The K-Means Algorithm described A is used for one iteration during which it seeks for a better clustering.

6) Mutation

The purpose of mutation which is the secondary search operator used in this work, is to generate modified individuals by introducing new features in the population. By mutation, the alleles of the produced child individuals have a chance to be modified, which enables further exploration of the search space. The mutation operator takes a single parameter pm, which specifies the probability of performing a possible mutation. Let $I = \{c_1, c_2,\ldots,c_k\}$ bean individual where each of whose gene ciis a cluster label. In our mutation operator, each gene *ci* is mutated through flipping this gene's allele from the current cluster label *ci* to a new randomly chosen cluster label if the probability test is passed. The mutation probability ensures that, theoretically, every region of the search space is explored. The mutation operator prevents the searching process from being trapped into local optima while adding to the diversity of the population and thereby in- creasing the likelihood that the algorithm will generate individuals with better fitness values.

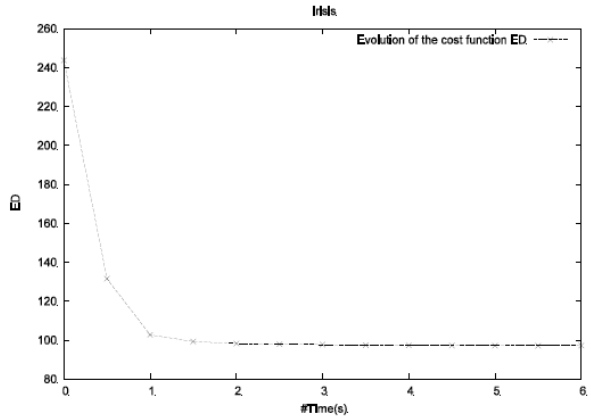

Fig. 1. Average development for 100 runs. Evolution of the cost function Eq. (1) for Irisis: Objects: 150. Attributes: 4.

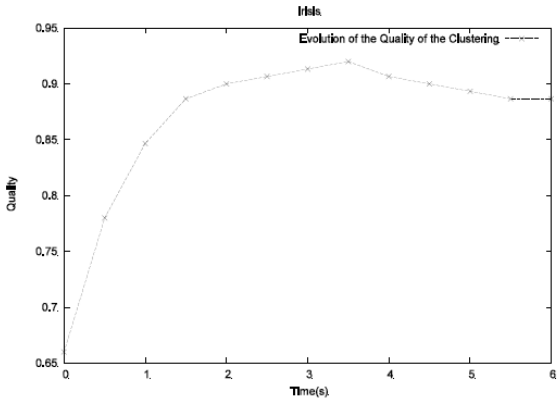

Fig. 2. Average development for 100 runs.Evolution of the quality of the clustering for Irisis.

7) Selection

The selection operator acts on individuals in the current population. During this phase, the search for the global solution gets a clearer direction, whereby the optimization process is gradually focused on therelevant areas of the search space. Based on each individual fitness, it determines the next population. In the roulette method, the selection is stochastic and biased to- wards the best individuals. The first step is to calculate the cumulative fitness of the whole population through the sum of the fitness of all individuals. After that, the probability of selection is calculated for each individual as being

$$P_{Selection} = \Phi / \sum_{i=1}^{|P|} \Phi_i$$

where | P | stands for the cardinality of the population.

8) Convergence criteria

As soon as the population tends to lose its diversity, convergence occurs and all individuals in the population tend to be identical with almost the same fitness value. The algorithm is assumed to reach convergence when no further improvement of the best solution has not been made during 5

consecutive generations.

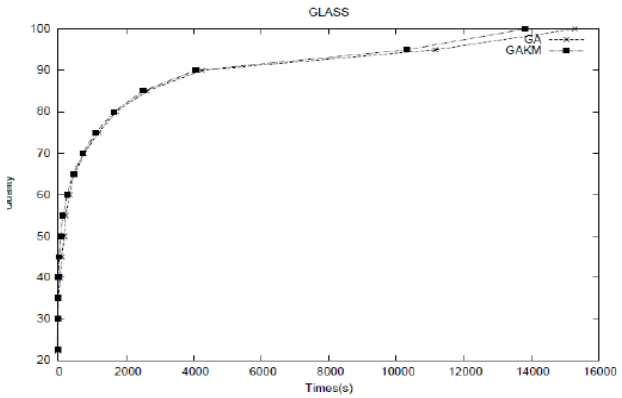

Fig. 3. GA Vs GAKM: GLASS: Objects: 214, Attributes: 9, Clusters: 6.

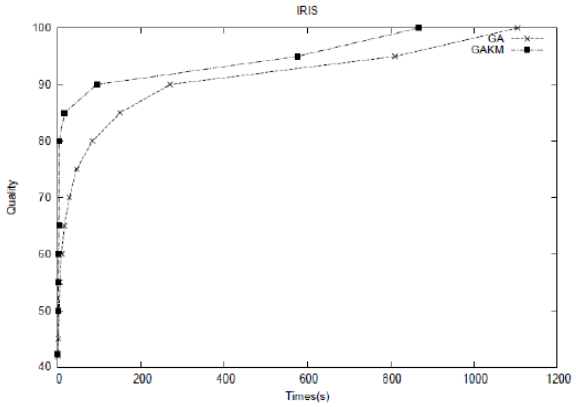

Fig. 4. GAKM: IRISIS: Objects: 150, Attributes: 4, Clusters: 3.

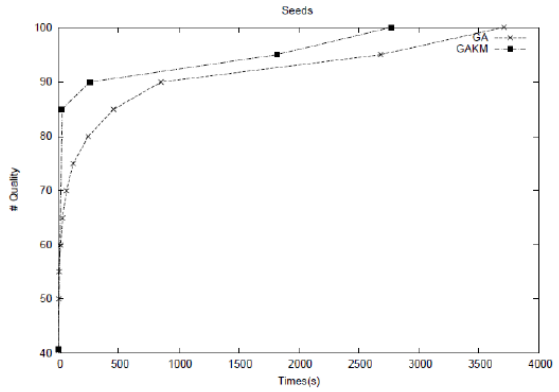

Fig. 5. GA Vs GAKM: SEEDS: Objects: 210, Attributes: 7, Clusters: 3.

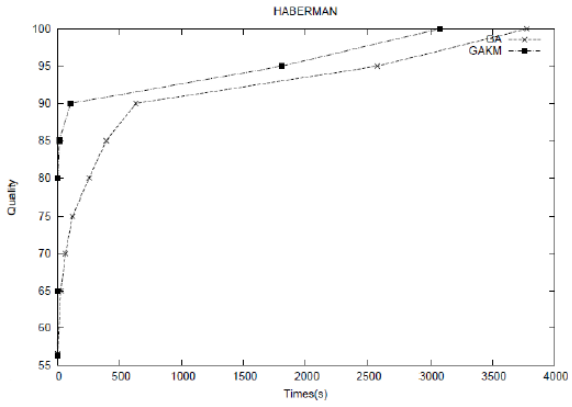

Fig. 6. GA Vs GAKM: HABERMANN: Objects: 306, Attributes: 3, Clusters: 2.

III. Benchmark Instances and Parameter Settings

The performance of the genetic algorithm is compared against its enhanced variant using a set of instances taken from real industrial problems. This set is taken from the Machine Learning Repository website (http://archive.ics.uci.edu/ml/datasets). Dueto the randomization nature of the algorithms, each problem instance was run 100 times. The tests were carried out on a DELL machine with 800 MHz CPU and 2 GB of memory. The code was written in C and compiled with the GNU C compiler version 4.6. The following parameters have been fixed experimentally and are listed below:

- Crossover probability = 0.85
- Mutation probability = 0.01
- Population size = 50

IV. Analysis of Results

The plot at Fig. 1 shows that the quality of the clustering is getting worse after reaching a peak at 0.91 while the plot in Fig. 2 shows that the cost function is indicatingthe opposite and attaining slightlylow values. This observation demonstrates that the cost function scores do not capture the quality of the clustering making it an unsuitable metric to apply for maximizing both the homogeneity within each cluster and the heterogeneity between different clusters. This is the main reason behind not comparing the value of ED produced by the two algorithms and rather choosing the solution of the clustering given in (http://archive.ics.uci.edu/ml/datasets) as the only criterion for comparing the quality of the clustering produced by GAKM and GA. The plot in Fig. 3-Fig. 6 show the evolution of the quality of the clustering as a function of time. Both algorithms start with random solutions leading to the same quality of clustering. During the first seconds of the search, GAKM already reaches high quality clusters (around 85%) in smaller amount compared to GA (89% of the time GA for Iris, 95% of the time of GA for Seeds, 4% of the time of GA for Glass, and finally 97% of the time of GA for Habermann). Both algorithms are able to avoid the so-called plateau regions. These regions span an area in the search space where the quality of the clustering remains unchanged. In all these four cases, the curve of GAKM remains above to that of GA and continue to maintain its superiority until convergence reached. Table I compares the running time of both algorithms. The two algorithms reach a success ratio of 100% but their times differ. The time spent does vary significantly with GAKM requiring the least amount of time (between 9% and 96% of the time of GA).

TABLE I: Comparing the time of GAKM and GA

Problems	**O**	**A**	**GAKM**	**GA**
GLASS	214	9	13.82s	35.0s
Habermann	306	3	3.26s	12.0s
Iris	150	3	0.86s	3.0s
Seeds	210	7	2.83s	20.0s
Tae	151	5	1.01s	3.0s
Blood	748	4	58.19s	200s
Bank	1372	4	401.69s	900s
Wine	178	13	1.86s	6.0s
Fertility	100	9	0.09s	1.0s

V. Conclusions

This paper introduces a combination of a genetic algorithm

and the popular K-Means for the clustering problem. The first conclusion drawn from the results at least for the in- stances tested in this work generally indicate that using K-Means as a strategy to improve the off-springs appears to improve substantially the performance of the genetic algorithm while producing the same quality of the clustering as GA. The quality of the clustering has always been compared against the solution given in (http://archive.ics.uci.edu/ml/datasets) and not through the value of the Euclidean Distance cost function widely used in literature since it does not capture the quality of the clustering making it an unsuitable metric to apply for maximizing both the homogeneity within each cluster and the heterogeneity between different clusters.

REFERENCES

[1] J. P. Bigus, *Data Mining with Neural Net- Works*, McGraw-Hill, 1996.

[2] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[3] G. Mecca, S. Raunich, and A. Pappalardo, "A new algorithm for clustering search results," *Data and Knowledge Engineering*, vol. 62, pp. 504-522, 2007.

[4] P. Baldi and S. Brunak, *Bioinformatics-The Machine Learning Approach*, MIT Press, 2001.

[5] P. M. B. Gerstein, "Integrative data mining: The new direction in bioinformatics-machine learning for analyzing genome-wide expression pro-files," in *Proc. the IEEE Engineering in Medicine and Biology*, vol. 20, pp. 33-40, 2005.

[6] G. Karypis, E. H. Han, and V. Kumar, "Chameleon: A hierarchical clustering algorithm using dynamic modeling," *IEEE Computer*, vol. 32, no. 8, vol. 6875, 1999.

[7] G. Karypis, E. H. Han, and V. Kumar, "Multilevel refinement for hierarchical cluster- ing. Technical Report TR-99-020," Department of Computer Science, University of Minnesota, Minneapolis, 1999.

[8] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document data-sets," in *Proc. the International Conference on Information and Knowledge Management*, 2002, pp. 515524.

[9] D. Boley, M. Gini, R. Gross, E. H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Partitioning- based clustering for web document categorization," *Decision Support Systems*, 1999.

[10] Y. Zhao and G. Karypis, "Comparison of Agglomerative and Partitional Document Clustering Algorithms," presented at SIAM Workshop on Clustering High-dimensional Data and Its Applications, 2002.

[11] S. Zhong and J. Ghosh, "A comparative study of generative models for document clustering," in *Proc. the International Conference on Data Mining Workshop on Clustering High Dimensional Data and Its Applications*, San Francisco, CA, May 2003.

[12] D. P. F. Alckmin and F. M. Varejao, "Hybrid genetic algorithm applied to the clustering problem," *Revista Investigation Operational*, vol. 33, no. 2, pp. 141-151, 2012.

[13] B. Juans and S. U. Guan, "Genetic algorithm based split-fusion clustering," *International Journal of Machine Learning and Computing*, vol. 2, no. 6, December 2012.

[14] L. A. N. Lorena and J. C. Furtado, "Constructive genetic algorithm for clustering problems," *Evolutionary Computation*, pp. 309-327, vol. 9 no. 3, September 2001.

[15] K. Adnan, A. Salwani, and N. M. Z. Ahmad, "A modified tabu search approach for the clustering problem," *Journal of Applied Sciences*, vol. 11, no. 19, 2011.

[16] D. O. V. Matos, J. E. C. Arroyo, A. G. D. Santos, and L. B. Goncalves, "A grasp based algorithm for efficient cluster formation in wireless sensor networks," in *Proc. the 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Commnications (WiMob)*, 2012, p. 187.

[17] J. B. MacQueen, "Some methods for classification and analysis of multivariate observation," in *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, Univ. of California Press, 1967.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

[19] D. Vrajitoru, "Genetic programming opera- tors applied to genetic algorithm," in *Proc. the Genetic and Evolutionary Computation Conference*, 1999, pp. 686-693.

[20] W. Spears, "Adapting crossover in evolutionary algorithms," in *Proc. the Fourth Annual Conference on Evolutionary Programming*, 1995, pp. 367-384.

[21] N. Bouhmala, "A Multilevel Memetic Algorithm for Large S at-Encoded Problems," *Evolutionary Computation*, vol. 20, no. 4, pp. 641-664, 2012.

[22] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, Arnold Publishers, 2001.

[23] M. R. Garey, D. S. Jhonson, and H. S. Witsenhausen, "The complexity of the generalized Llyod-Max problem," *IEEE Trans Info Theory*, vol. 28, no. 2, pp. 255-256, 1982.

[24] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," *Supercomputing*, San Diego, 1995.

[25] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput*., vol. 20, no. 1, pp. 359-392, 1998.

[26] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *J. Par. Dist. Comput*., vol. 48, no. 1, pp. 96-129, 1998.

[27] L. L. Kaufman and P. J. Rousseuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York, 1990.

[28] C. Walshaw, *Multilevel Refinement for Combinatorial Optimization: Boosting Meta-heuristic Performance*, pp. 261-289, Springer, Berlin, Germany, 2008.

Noureddine Bouhmala is an associate professor at Buskerud and Vestfold University College. His research interests include data mining, meta-heuristics, Combinatorialoptimizaton**.**

Anders Viken is a computer engineer who graduated from Buskerud and Vestfold University College in 2014. He is currently studying systems engineering at Buskerud and Vestfold University College, while working part time as project engineer at Kongsberg Maritime AS.

Jonas Blåsås Lønnum is a computer engineer who graduated from Buskerud and Vestfold University College in 2014. He is currently working as a software developer at Ablemagicin Trondheim.