Metaheuristic Optimization for Automatic Clustering of Customer-Oriented Supply Chain Data

César L. C. Mattos, Guilherme A. Barreto
Federal University of Ceará (UFC), Campus of Pici
Department of Teleinformatics Engineering
Center of Technology, Fortaleza, Ceará, Brazil
cesarlincoln@terra.com.br, gbarreto@ufc.br

Dennis Horstkemper, Bernd Hellingrath
Westfälische Wilhelms-Universität (WWU-) Münster
Institut für Wirtschaftsinformatik - ERCIS
Leonardo Campus 3, 48149, Münster, Germany
{dennis.horstkemper, bernd.hellingrath}@ercis.uni-muenster.de

***Abstract*—In this paper we evaluate metaheuristic optimization methods on a partitional clustering task of a real-world supply chain dataset, aiming at customer segmentation. For this purpose, we rely on the automatic clustering framework proposed by Das *et al.* [1], named henceforth DAK framework, by testing its performance for seven different metaheuristic optimization algorithm, namely: simulated annealing (SA), genetic algorithms (GA), particle swarm optimization (PSO), differential evolution (DE), artificial bee colony (ABC), cuckoo search (CS) and fireworks algorithm (FA). An in-depth analysis of the obtained results is carried out in order to compare the performances of the metaheuristic optimization algorithms under the DAK framework with that of standard (i.e. non-automatic) clustering methodology.**

I. Introduction

Nowadays, supply chain managers need to cope with constantly more complex network structures and ever more demanding customers. Therefore, they have to ensure cost optimal operations while fulfilling the needs of vastly different customers, who in turn have alternating expectations in terms of product design, costs and services. Within this field of tension, practitioners and academics alike came to the conclusion, that the traditional "one size fits all" philosophy of designing supply chains and their processes is not sufficient anymore. Following this approach, several customer needs would be under- or over-served [2]. Instead, separable logistical processes and supply chain setups should exist for the most viable customer groups.

However, research as well as industrial practice regarding the identification of such customer groups is nowadays still mostly focused on simple heuristics, that only differentiate by one or two characteristics (variables) [3], [4], [5]. Furthermore, they often make use of domain knowledge, which in turn might limit the identification of newly emerging groups with coherent desires [6]. As companies begin to compete more and more through the offered logistical services instead through products and prices alone, it is nevertheless important to increase the viability of the to-be-identified customer groups in terms of the homogeneity of their requirements in a more generic manner.

A promising alternative for the identification of different ecological and logistical viable customer groups are data clustering algorithms, as suggested in a few related works [7], [8]. These studies have shown that partitional clustering algorithms, particularly the K-means [9] and the SOM [10], can identify such customer groups efficiently without relying on predefined rules and while employing 3 or more variables. However, the major drawback of this standard clustering methodology, is that the number of clusters must be specified in advance. If this number is not available (e.g. by a priori domain knowledge), it must be searched for with the help of exhaustive and repetitive application of cluster validity indices [11].

978-1-5090-6638-4/17/$31.00 ©2017 IEEE

In this paper, we explore an alternative approach where the number of cluster need not to be defined beforehand, but rather it is included as a parameter to be optimized *on the fly* as part of the clustering process. This is the main idea underlying automatic partitional clustering strategies [12], which are capable of simultaneously finding the appropriate number of clusters and the corresponding centroids. For this purpose, we adopt the methodology proposed in [1], named henceforth the Das-Abraham-Konar (DAK) framework, to perform automatic clustering using several metaheuristic optimization algorithms. The DAK framework is evaluated in a real-world scenario by means of a challenging dataset obtained from consumer packaged goods industry. Obtained results are very promising, strongly suggesting the DAK framework as a viable alternative to the standard (i.e. non-automatic) clustering methodology.

The remainder of the paper is organized as follow. In Section II we review basic concepts of partitional clustering, some cluster validity indices and the K-means algorithm. In Section III we describe the automatic metaheuristic clustering framework we apply to the problem of interest. The evaluated metaheuristic optimization methods are described in Section IV. The dataset, its industrial context, and the variables chosen for the clustering problem are presented in Section V. The results of the comprehensive performance evaluation are presented and discussed in Section VI. The paper is concluded in Section VII.

II. The Basics of Partitional Clustering

Let $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, where $\boldsymbol{x}_n \in \mathbb{R}^D$, be a set of N D-dimensional patterns. The clustering problem is an unsupervised task whereby the mentioned set is partitioned into K clusters $\mathcal{C} = \{C_k\}_{k=1}^{K}$ of elements in such a way that patterns within the same cluster tend to have high similarity and patterns from different clusters should have low similarity. These properties may be measured by specific user-defined objective function [9].

Besides this intuitive notion for clustering, the obtained partitions must not be empty nor overlap with other partition and together they must form the original set $\mathcal{X}$. Thus, the clustering task may be stated as an optimization problem:

$$
\begin{aligned}
\underset{\mathcal{C}}{\text{Optimize}} \quad & f(\mathcal{X}, \mathcal{C}), \\
\text{s.t.} \quad & C_k \neq \emptyset, \quad \forall k, \\
& C_k \cap C_{k'} = \emptyset, \quad \forall k \neq k', \\
& \bigcup_{k=1}^{K} C_k = \mathcal{X},
\end{aligned}
\tag{1}
$$

where $f(\cdot)$ is the previously mentioned objective function.

A. Clustering Cost Functions

The clustering objective function $f(\cdot)$ in Eq. (1), sometimes called a *clustering validity index*, evaluates a given set of partitions $\mathcal{C}$ as a proper clustering for the set of patterns $\mathcal{X}$ considering the following aspects:

- **Cohesion (or compactness)**: Patterns in the same cluster should be as similar as possible;
- **Separation**: Clusters should be well separated from each other.

While many cluster validation indices are available elsewhere (see e.g. [12], [9]), in this paper we rely only on the Davies-Bouldin (DB) and Pakhira-Bandyopadhyay-Maulik (PBM) as clustering objective functions. The former is a classic and very popular metric for cluster validation, while the latter is a index that has been applied more recently in the literature. Both indexes may be found in standard scientific programming tools, such as Matlab, Octave and R.

Davies-Bouldin (DB) index - The DB index is usually calculated by [13]:

$$
\mathrm{DB}(\mathcal{C}) = \frac{1}{K} \sum_{k=1}^{K} \max_{k \neq k'} \left(\frac{\delta_k + \delta_{k'}}{\Delta_{kk'}} \right), \tag{2}
$$

where δ_k and $\Delta_{kk'}$ are respectively called *within cluster scatter* and *between cluster distance* and are defined as

$$
\begin{aligned}
\delta_k &= \frac{1}{N_k} \sum_{\boldsymbol{x}_n \in C_k} \|\boldsymbol{x}_n - \boldsymbol{m}_k\|, & (3) \\
\Delta_{kk'} &= \|\boldsymbol{m}_k - \boldsymbol{m}_{k'}\|, & (4) \\
\boldsymbol{m}_k &= \frac{1}{N_k} \sum_{\boldsymbol{x}_n \in C_k} \boldsymbol{x}_n,
\end{aligned}
$$

where $\|\cdot\|$ denotes the standard Euclidean norm, N_k is number of elements in partition C_k and $\boldsymbol{m}_k$ is its *centroid.*

The DB index praises solutions which equilibrate compact clusters and separated centroids in the worst scenario for each partition (small values of $\Delta_{kk'}$). Good clustering solutions results in lower DB index values, i.e., as an objective function, it must be minimized.

Pakhira-Bandyopadhyay-Maulik (PBM) index - The PBM index, proposed in [11], is defined as

$$
\mathrm{PBM}(\mathcal{C}) = \left(\frac{D_B}{K} \frac{E_T}{E_W} \right)^2, \tag{5}
$$

where

$$
D_B = \max_{k \neq k',\, k, k' \in [1:K]} \Delta_{kk'}, \quad E_W = \sum_{k=1}^{K} N_k \delta_k \tag{6}
$$

and

$$
E_T = \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{m}\|, \quad \boldsymbol{m} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n, \tag{7}
$$

with $\boldsymbol{m}$ standing for the centroid of the whole data. The PBM index favors fewer and more compact clusters. Good clustering solutions lead to greater PBM index values. Thus, as an objective function, it must be maximized or, equivalently, its negative value must be minimized.

B. The K-Means algorithm

The famed K-Means (KM) clustering algorithm has a very simple objective function:

$$
KM(\mathcal{C}) = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_n \in C_k} \|\boldsymbol{x}_n - \boldsymbol{m}_k\|^2, \tag{8}
$$

which is sometimes called the *quantization error* function. The steps of the associated iterative algorithm, given a fixed number K of clusters and some initial set of partitions $\mathcal{C}_0$, are given next.

1) Find all cluster partitions C_k, $k = 1, \ldots, K$:

$$
C_k = \{\boldsymbol{x}_n \in \mathbb{R}^D \mid \|\boldsymbol{x}_n - \boldsymbol{m}_k\| < \|\boldsymbol{x}_n - \boldsymbol{m}_j\|, \forall j \neq k\}.
$$

2) Recompute the cluster centroids:

$$
\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x}_n \in C_k} \boldsymbol{x}_n, \, \forall k.
$$

These two steps are repeated iteratively until the centroids positions do not present relevant changes.

Although being simple and fast, the KM algorithm is very sensitive to initial conditions. Furthermore, it requires the *a priori* definition of the number K of clusters. Thus, the standard partitional clustering methodology involves the combined use of the KM algorithm (or similar algorithm, such as the SOM) and cluster validation indices (e.g. DB or PBM). For each $K \in [2, K_{max}]$, where K_{max} is the maximum allowed number of clusters, one has to compute the corresponding value of the chosen index. The suggested optimal number of cluster is the one that minimizes (or maximizes) the chosen index. This process should be repeated several times in order to reduce the effects of centroid initialization.

As an alternative, automatic clustering strategies, in which the number of clusters need not to be defined in advance, but rather included as part of the optimization process, have been proposed. We describe one of such approaches, developed for metaheuristic optimization techniques, in the next section.

III. The DAK Framework

Considering initially a population-based metaheuristic optimization algorithm, let the i-th candidate solution in the t-th iteration to be represented by the vector $\boldsymbol{z}_i(t)$, defined as

$$\boldsymbol{z}_i(t) = [T_{i,1}\ T_{i,1}\ \cdots\ T_{i,K_{\max}} \mid \boldsymbol{m}_1^{(i)}\ \boldsymbol{m}_2^{(i)}\ \cdots\ \boldsymbol{m}_{K_{\max}}^{(i)}], \quad (9)$$

where $T_{i,k} \in [0,1]$ is the activation threshold of the k-th cluster, $\boldsymbol{m}_k^{(i)}$ is the k-th cluster centroid and $K_{\max}$ is the maximum allowed number of clusters. It should be noted that $\boldsymbol{z}_i(t)$ is a Q-dimensional vector, where $Q = K_{\max}(1+D)$ and D is the dimension of the pattern vectors.

The activation thresholds $T_{i,k}$ define which clusters (out of K_{max}) are active; in other terms, which of the centroids $\boldsymbol{m}_k^{(i)}$ are to be considered to partitioning the patterns. A cluster is active only if $T_{i,k} \geq 0.5$. Otherwise, it is ignored during partitioning. Once the metaheuristic optimization process converges, the optimal number of clusters K_{opt} is the number of remaining active clusters.

In order to guarantee that all candidate solutions $\boldsymbol{z}_i(t)$ are feasible solutions, the following the control procedures are adopted: (i) If there are less than two active clusters, randomly choose two activation thresholds and set them to random values between 0.5 and 1. (ii) If there are empty or single-element active clusters, assign the centroids $\boldsymbol{m}_k^{(i)}$ to the provided KM solution with K equal to the number of active clusters.

The DAK framework has been developed aimed at the DE algorithm, but it was further tested with the PSO and GA algorithms. In the present paper we aim at enlarging the set of tested metaheuristic optimization methods under the DAK framework, including other population-based algorithms, such as the artificial bee colony, and trajectory-based techniques as well, such as the simulated annealing, cuckoo search and fireworks algorithms. All the metaheuristics mentioned in this paragraph will be briefly described in the next section.

IV. Evaluated Metaheuristic Algorithms

The formulation of the candidate solution in Eq. (9) is general enough to be used by any metaheuristic algorithm of choice. In this section we briefly describe the main equations of the seven algorithms chosen to tackle our automatic clustering problem. Our choice of algorithms contains both classical and more recent algorithms. Among the classical ones, we mention the SA algorithm, which is a trajectory-based algorithm, since it provides just one candidate solution at each iteration. GA, PSO and DE can nowadays be considered classical methods. The remaining algorithms (ABC, CS and FA) are more recent population-based metaheuristics.

A. Simulated Annealing

The SA algorithm [14] is a trajectory-based optimization technique developed from a computational metaphor inspired by the annealing process in metallurgy. By means of initial heating and controlled cooling it is possible to modify the physical properties of a material to increase its ductility and reduce its hardness, making it more workable. The metaphor works as follows: the current state of the material's structure plays the whole of the candidate solution, the quality of the candidate solution is then evaluated by the objective function to be optimized, which defines the optimization surface. A temperature parameter emulates the heating/cooling process, allowing the candidate solution to escape from local minima as time goes by.

At each iteration t, the candidate solution $\boldsymbol{z}(t)$ is adapted through the following expressions[1]:

$$\boldsymbol{z}(t+1) = \begin{cases} \boldsymbol{z}'(t), & \text{if } P_a \geq \mathrm{U}(0,1), \\ \boldsymbol{z}(t), & \text{otherwise.} \end{cases}, \quad (10)$$

with

$$\boldsymbol{z}'(t) = \boldsymbol{z}(t) + \eta_{SA}(\boldsymbol{z}_{\max} - \boldsymbol{z}_{\min})\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_Q) \quad (11)$$

where η_{SA} is an incremental step size, $\boldsymbol{z}_{\max}$ and $\boldsymbol{z}_{\min}$ are respectively the maximum and the minimum possible values for each element of the vector $\boldsymbol{z}$. The acceptance probability is computed as

$$P_a = \min\left\{1, \exp\left(\frac{f(\boldsymbol{z}(t)) - f(\boldsymbol{z}'(t))}{\tau(t)}\right)\right\}, \quad (12)$$

where $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_Q)$ is a Q-dimensional vector sampled from a zero-mean multivariate normal distribution with identity covariance matrix, $\mathrm{U}(0,1)$ is a uniformly distributed random number between 0 and 1 and $\tau(t)$ is the value of the temperature parameter, which must decay along the iterations.

B. Genetic Algorithm (GA)

The class of GA was introduced in [15] as a computational metaphor to the modern synthesis of the Darwinian evolution by natural selection with genetics. As such, iterations t are called *generations*, candidate solutions $\boldsymbol{z}_i(t)$ become *chromosomes* and their components are *genes*. The algorithm searches for good solutions by successive applications of basic *genetic operators*: selection, crossover and mutation[2]. Its basic formulation follows the steps below:

GA.1 - Selection of mating chromosomes. We use the *binary tournament selection*, as described in [16].

GA.2 - Apply the crossover operator to the selected mates with probability P_r. We choose the following SBX (Simulated Binary Crossover) operator, as described in [17]:

$$\begin{aligned} z_{aq}(t) =& \frac{1}{2}[(1+\gamma_q)z_{iq}(t)(1-\gamma_q)z_{jq}(t)], \quad \forall q, \\ z_{bq}(t) =& \frac{1}{2}[(1-\gamma_q)z_{iq}(t)(1+\gamma_q)z_{jq}(t)], \quad \forall q, \\ \gamma_q =& \begin{cases} (2R_q)^{\frac{1}{2}}, & \text{if } R_q \leq 0.5 \\ \left(\frac{1}{2(1-R_q)}\right)^{\frac{1}{2}}, & \text{otherwise.} \end{cases}, \quad \forall q, \end{aligned}$$

where $\boldsymbol{z}_i(t)$ and $\boldsymbol{z}_j(t)$ are the selected parents, $\boldsymbol{z}_a(t)$ and $\boldsymbol{z}_b(t)$ are the produced offspring and $R_q \sim \mathrm{U}(0,1)$. When the crossover is not applied, we simply copy the parents to the offspring.

[1]In this paper we will denote the multiplication of two vectors element-wise (Hadamard product) as traditional product between scalars, as in Eq. (11), to avoid unnecessary notation clutter.

[2]Sometimes, elitism is added to this set of operators.

GA.3 - Apply the mutation operator to the offspring with probability P_m. Th mutation operator is the same as in Eq. (11), with a different step η_{GA}.

GA.4 - Built the population for the next generation. In this paper we apply $(\mu + \lambda)$-selection, as described in [17].

C. Particle Swarm Optimization (PSO)

The PSO algorithm [18] is inspired by social behavior and self-organization observed, for instance, in flocks of migratory birds or fish schools. The exchange of information between the elements of the population promotes exploration, while the individuals' self experience is responsible for the exploitation. At each generation, the i-th candidate solution $\boldsymbol{z}_i(t)$, called *particle*, is used to update its velocity vector as

$$\begin{aligned} \boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + \boldsymbol{\phi}_1(\boldsymbol{z}_{\text{pbest},i}(t) - \boldsymbol{z}_i(t)) \\ &\quad + \boldsymbol{\phi}_2(\boldsymbol{z}_{\text{lbest},i}(t) - \boldsymbol{z}_i(t)), \end{aligned} \tag{13}$$

where w is an inertia coefficient, $\boldsymbol{\phi}_j = c_j \mathrm{U}(\mathbf{0}_Q, \mathbf{1}_Q)$, $j = 1, 2$, with $\mathrm{U}(\mathbf{0}_Q, \mathbf{1}_Q)$ denoting a Q-dimensional random vector uniformly distributed between 0 and 1; c_1, c_2 are acceleration coefficients, $\boldsymbol{z}_{\text{pbest},i}(t)$ is the best solution found by the i-th particle so far, and $\boldsymbol{z}_{\text{lbest},i}(t)$ is the best solution found among the particles in its neighborhood so far.

Finally, the corresponding position vector of the i-th particle is updated as follows:

$$\boldsymbol{z}_i(t+1) = \boldsymbol{z}_i(t) + \boldsymbol{v}_i(t+1). \tag{14}$$

D. Differential Evolution (DE)

Simply put, the DE algorithm [19] is population-based evolutionary algorithm that also uses genetic operators as GAs, but applied in the reverse order (i.e. mutation $\rightarrow$ crossover $\rightarrow$ selection). At generation t, a mutated version of the i-th candidate solution $\boldsymbol{z}_i(t)$, $i = 1, \ldots, N$, is generated as follows:

$$\boldsymbol{s}_i(t) = \boldsymbol{z}_{r_1}(t) + \eta_{DE}(\boldsymbol{z}_{r_2}(t) - \boldsymbol{z}_{r_3}(t)), \tag{15}$$

where r_1, r_2 and r_3 are randomly sampled indexes of chromosomes other than i and η_{DE} is a scale factor. Then, the crossover operator is applied component-wisely to the candidate solution $\boldsymbol{z}_i(t)$ and its mutated version $\boldsymbol{s}_i(t)$:

$$z'_{i,q}(t) = \begin{cases} s_{i,q}(t), & \text{if } \mathrm{U}(0,1) < P_r, \\ z_{i,q}(t), & \text{otherwise.} \end{cases}, \; q = 1, ..., Q, \tag{16}$$

where P_r is the crossover rate. Finally, assuming a minimization problem, the offspring $\boldsymbol{z}'_i(t)$ replaces or not the current candidate solution $\boldsymbol{z}_i(t)$ in the next population according to the following rule:

$$\boldsymbol{z}_i(t+1) = \begin{cases} \boldsymbol{z}'_i(t), & \text{if } f(\boldsymbol{z}'_i(t)) < f(\boldsymbol{z}_i(t)), \\ \boldsymbol{z}_i(t), & \text{otherwise.} \end{cases} \tag{17}$$

E. Artificial Bee Colony (ABC)

The ABC algorithm [20] is inspired by the intelligent foraging behavior of honey bee swarms when searching for food sources. We describe next a more recent version of ABC, as proposed in [21]. The artificial model contains three groups of individuals: employed bees, onlookers and scouts. The hive works in order to find good food sources (candidate solutions, $\boldsymbol{z}_i$) and to improve the quality (objective function) of the known sources. The algorithm steps are executed as follows:

ABC.1 - Send employed bees to search for different food sources. Each food source is then exploited through the expression:

$$\boldsymbol{z}_i(t+1) = \begin{cases} \boldsymbol{z}'_i(t), & \text{if } f(\boldsymbol{z}'_i(t)) < f(\boldsymbol{z}_i(t)), \\ \boldsymbol{z}_i(t), & \text{otherwise.} \end{cases} \tag{18}$$

where the q^*-th component of $\boldsymbol{z}'_i(t)$ is given by

$$z'_{i,q^*}(t) = z_{i,q^*}(t) + \mathrm{U}(0,1)(z_{i,q^*}(t) - z_{a,q^*}(t)), \tag{19}$$

where $q^* \in [1, Q]$ is the index of a randomly chosen component of the vector $\boldsymbol{z}_i(t)$ and a is the index of a randomly chosen candidate solution other than i. Each time Eq. (18) is evaluated without improvement, a counter associated with the i-th source is incremented. When an improvement is achieved, such counter is reseted to zero.

ABC.2 - Assign randomly onlooker bees to the food sources with probability proportional to their quality. Those bees also perform exploitation in their food sources via Eqs. (18)-(19).

ABC.3 - Abandon the worst exhausted food source and replace it with a new random candidate solution. A source is considered exhausted when its counter is greater than some previously defined threshold.

F. Cuckoo Search (CS)

The CS algorithm [22] aims at imitating the curious parasite behavior of some cuckoo species, which lay eggs in the nest of other host non-cuckoo birds. In order to avoid the host bird from discovering the alien egg and abandon it, the female cuckoo alter the appearance of its own egg for it to resemble the original eggs of the host. We follow more closely the implementation presented in [23], summarized by the steps below:

CS.1 - Update each host nest (candidate solution, $\boldsymbol{z}_i$) through a Lévy flight:

$$\boldsymbol{z}_i(t+1) = \begin{cases} \boldsymbol{z}'_i(t), & \text{if } f(\boldsymbol{z}'_i(t)) < f(\boldsymbol{z}_i(t)), \\ \boldsymbol{z}_i(t), & \text{otherwise.} \end{cases}, \tag{20}$$

so that

$$\boldsymbol{z}'_i(t) = \boldsymbol{z}_i(t) + \eta_{CS}\text{Lévy}(\lambda)(\boldsymbol{z}_{\text{best}}(t) - \boldsymbol{z}_i(t)), \tag{21}$$

where η_{CS} is an update stepsize, $\boldsymbol{z}_{\text{best}}(t)$ is the best solution found so far and $1 < \lambda \leq 3$ is the parameter of the Lévy distribution. Such distribution presents heavy tails, which allow for eventual longer stepsize lengths.

CS.2 - Perform the following randomization in a randomly selected fraction P_a of the nests:

$$\boldsymbol{z}_i(t+1) = \begin{cases} \boldsymbol{z}'_i(t+1), & \text{if } f(\boldsymbol{z}'_i(t+1)) < f(\boldsymbol{z}_i(t+1)), \\ \boldsymbol{z}_i(t+1), & \text{otherwise.} \end{cases}, \tag{22}$$

so that

$$\boldsymbol{z}'_i(t+1) = \boldsymbol{z}_i(t+1) + \mathrm{U}(0,1)(\boldsymbol{z}_a(t+1) - \boldsymbol{z}_b(t+1)), \tag{23}$$

where a and b are randomly chosen indexes of nests.

G. Fireworks Algorithm (FA)

The FA algorithm [24] is inspired by the observation of a firework being set off, when a shower of sparks fill a local area around the explosion, which could resemble a series of local search procedures. The algorithm works as follows:

FA.1 - Set off fireworks at each location (candidate solution) $z_i(t)$. Therefore, calculate the number of generated sparks, the amplitude of their explosion and their location. Those regular sparks are uniformly distributed around the firework. The detailed procedure can be found in [24].

FA.2 - Select a random fraction of fireworks to generate a specific spark, in order to maintain the diversity of the sparks. As the original paper, we also opted for Gaussian distributed sparks.

FA.3 - Keep the best location found and randomly select a fraction of all the other locations (fireworks and sparks) proportionally to the quality of their location. Those locations are carried to the next iteration.

V. A Real-World Supply Chain Dataset

In order to assess the applicability of the DAK framework to the customer segmentation problem we are interested in, we perform a case study with data from an enterprise positioned on the consumer packaged goods industry. This enterprise, which operates in 17 European countries and possesses distribution centers in five of them, employs about 2000 persons and sells its products in 40 different markets. In several of these markets, the enterprise is the current market leader. Fig. 1 shows an excerpt of the enterprises supply chain. It comprises several facilities, such as the enterprises own production plant for premium table-top products and packaging solutions for take-away food. The enterprise is sourcing its paper from a local paper mill. It furthermore operates one international and four regional distribution centers allowing for efficiently bundled transports. An external supplier, who is providing additional products sold under the enterprises brand name is also a member of this supply chain.

Currently, 88% of the products sold by the enterprise are manufactured following the make-to-stock principle and only 12% are manufactured following the make-to-order principle. Thus, the overall costs observed in the supply chain are caused due to a large part through high inventory amounts. Interviews with the enterprises decision makers revealed, that one of their new strategic goals is to apply a customer-oriented supply chain segmentation. Their main target is an overall reduction of inventory amounts throughout their overall supply chain. The first step towards performing such a supply chain segmentation is to perform a customer segmentation to identify valuable customer groups, which justify introducing corresponding and distinct supply chain segments.

The final customers served by the enterprise consist of retailers such as grocery stores, supermarkets, specialty retailers and interior design stores, as well as specialists, such as hotels, restaurants and catering companies. Therefore, it can be expected, that this diverse set of customers will have significantly different characteristics which could be better served by appropriately differentiated supply chain configurations. While one could be tempted to simply differentiate the groups of retailers and specialists, investigating further characteristics of the different customers might lead to the creation of much more logistical viable customer segments. Thus, the application of quantitative clustering methods is a promising approach for this enterprise to pursue its strategic goals.

In general, demands in the industry of consumer packaged goods are characterized by hard to predict and vastly varying demand fluctuations [25]. This also holds true for the investigated enterprise and further motivates the application of a customer-specific supply chain segmentation. For this proof-of-concept case study, we will from here on focus on a specific product sold by the enterprise. This product is sold to the highest amount of individual customers of the enterprise (1,701), belongs to the most revenue generating products of the enterprise and also specifically showcases high demand variations. The enterprise currently has 11 variables (x_1 to x_{11}) available to characterize specific customers for this product, which are described next.

The *accepted delivery time* (x_1) of a customer identifies the time window which can be used to satisfy customer demands after a specific order has arrived. The *annual revenue per customer* (x_2) is resulting from the operational costs and the turnover of every customer per year. Here, the costs of supply chain operations are composed of costs for storage (interest and stocking), production, procurement and transportation. This variable can be used to measure a customers importance to the enterprise. Furthermore, the *price willingness* (x_3) describes the maximum prize that a customer is willing to pay. This information could be used to determine the highest possible logistics service level that could be realized within the price conception of a customer.

For measuring the currently offered logistics service level, first the *delivery reliability* (x_4) for a customer can be used. For the enterprise of interest achieving a delivery reliability of at least 90% is one of the most important business objectives. The variable itself is calculated by using two further variables, namely: *the total amount of goods delivered in time* (x_5) and *the total amount of goods ordered by one customer* (x_6). The delivery reliability is a valuable source of information for identifying customer groups which do not experience the intended minimal delivery reliability. Measures like an extended forecast-driven part of the supply chain, a higher safety stock level or a dedicated value chain processes could improve the delivery reliability and finally increase customer satisfaction.

The *standard deviation of demand* (x_7) describes the relation of the forecasted order amounts to the real order amounts of a customer and can be used as a baseline to analyze the customers demand volatility. A further variable for describing the volatility of customer demands is *variation coefficient of order amounts* (x_8). This variable is calculated by the standard deviation of demand in relation to the average demand of the customer during the planning horizon. Customers with a more constant variation coefficient of orders allow for measures like a broadening of the order-driven area of a supply chain or a decrease of safety stock levels. Instead of this, customers who have a higher variation coefficient of orders need an

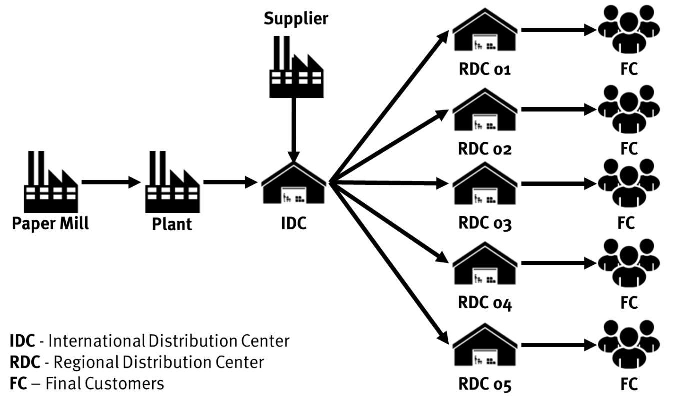

Fig. 1. Supply chain of investigated enterprise.

extended forecast-driven area or higher safety stock levels to avoid stock-outs, which are more likely in this case.

Another view on the demand volatility can be taken by considering the *total number of orders* (x_9), which describes the number of orders a customer performed within a planning horizon, and the *average number of orders* (x_{10}), defining the typical amount of orders a customer has performed peer planning horizon over the timespan of a whole year. The *standard deviation of orders* (x_{11}) sets these variables in relation to each other. This differentiation from the order quantities is important, as unpredicted orders from customers might need to be transported via different transport modes and cause additional costs, e.g. through an express delivery from the enterprises production plant instead of using the usual bundled transports through the various distribution centers. This in turn means that a high standard deviation of orders is another driver for an increased amount of safety stocks, while a low value of this variable would allow for a decrease of such stocks.

As could be expected, several of these variables are linearly dependent. Therefore, a meaningful selection of variables has to be chosen. Interviews with the industrial partner showed, that they are particularly interested in the relation between delivery reliability, costs and revenues for the different customer segments. Therefore, the further analysis will focus on the variables x_1, x_2, x_4, x_7 and x_8.

VI. Experimental Results and Discussion

We evaluated the seven metaheuristic algorithms described in Section IV in the task of automatic clustering of the logistic data presented in Section V. All population-based methods were executed for a total of $N_{\text{iter}} = 1000$ iterations, while the SA algorithm was executed for $N_{\text{iter}} = 10000$ iterations. The variables were rescaled to the range $[0, 1]$. The experiments were prepared and ran in the R language [26]. Most values of the parameters for all metaheuristics were taken from their original papers while some were found empirically after some initial experiments. A summary of the used parameters are in Table I.

As a baseline for the performance comparison, we consider the KM algorithm. Since it requires the *a priori* definition of the number of clusters, we run it 100 times for each value of $K = 1, \ldots, K_{max}$ and select the solution resulting in the lowest quantization error for each K. The DB and PBM indexes are chosen for cluster validation purposes, with the caveat that the first should be minimized, while the second must be maximized. Both indexes were calculated via the R package *clusterCrit* [27]. The maximum number of clusters allowed was $K_{\text{max}} = 10$.

TABLE I. Summary of parameters for all the evaluated metaheuristic algorithms.

	Parameter	Value
SA	Initial Incremental Step ($\eta_{SA}(1)$)	0.1
	Initial Incremental Step ($\eta_{SA}(N_{\text{iter}})$)	0.001
	Maximum Temperature	1000
	Exponential Temperature Decay Factor	0.98
GA	Population Size	100
	Crossover Rate (P_r)	0.95
	Mutation Rate (P_m)	0.05
	Mutation Step (η_{GA})	0.01
PSO	Population Size	40
	Acceleration Coefficients (c_1, c_2)	1.193
	Inertia Coefficients (w)	0.721
DE	Population Size	100
	Crossover Rate (P_r)	0.9
	Scale factor (η_{DE})	0.8
ABC	Population Size	40
	Limit for Counter of Exploitations	$40Q$
CS	Population Size	25
	Probability for Randomization Step	0.25
	Update Step (η_{CS})	1
FA	Population Size	5
	Maximum Number of Sparks	50
	Number of Gaussian Sparks	5
	Maximum Amplitude of Explosion	40

Besides the DB and PBM indexes, we also report the *a posteriori*[3] values of the *mean within cluster scatter* (MWCS) and the *mean between cluster distance* (MBCD), respectively defined as follows:

$$\text{MWCS} = \frac{1}{K}\sum_{k=1}^{K}\delta_k, \qquad (24)$$

$$\text{MBCD} = \frac{1}{2(K-1)}\sum_{k=1}^{K}\sum_{k'=1}^{K}\Delta_{kk'}, \qquad (25)$$

where δ_k and $\Delta_{kk'}$ were defined in Eqs. (3) and (4). It should be noticed that lower values for MWCS and greater values for MBCD are preferable.

In Tables II and III we report the best results obtained through optimization guided by DB and PBM indexes, respectively. The number of elements in each partition found is also presented in their last columns. Before proceeding with the analysis of the results, a word of caution is in order. Cluster analysis of real-world data must be done with extreme care, because there may not be a single, definitive solution. Many options should be analyzed. Technical knowledge as represented by the clustering methods evaluated here must be balanced with domain knowledge of the supply chain management (SCM) field.

If one compares the results reported in both tables, the results with the optimization of the PBM index (Table III) are much more homogeneous. In Table II there are four different suggested values for K_{opt} ({3,5,6,7}), while in Table III we

[3]That is, computed only after the clusters have been found by the proposed methodology.

TABLE II. SUMMARY OF AUTOMATIC CLUSTERING RESULTS WHEN OPTIMIZING THE DB INDEX, ORDERED BY BEST INDEX VALUE FOUND.

	DB	PBM	MWCS	MBCD	Partitions
DE	**0.1265**	1.1947	0.0659	**1.0144**	7: [572, 414, 359, 270, 78, 5, 3]
CS	0.1309	1.4430	0.0640	0.8169	6: [571, 394, 262, 261, 211, 2]
PSO	0.1325	1.3634	0.0668	0.9399	6: [571, 448, 398, 264, 16, 4]
ABC	0.1337	1.2254	**0.0595**	0.6955	6: [572, 427, 256, 242, 202, 2]
FA	0.1390	1.5497	0.0703	0.8081	6: [572, 335, 267, 254, 211, 62]
GA	0.1428	**2.2092**	0.0673	0.8635	5: [571, 463, 399, 266, 2]
SA	0.1451	1.7526	0.0716	0.7900	6: [572, 275, 270, 223, 181, 180]
KM	0.2510	1.8556	0.1041	0.5164	3: [643, 575, 483]

TABLE III. SUMMARY OF AUTOMATIC CLUSTERING RESULTS WHEN OPTIMIZING THE PBM INDEX, ORDERED BY BEST INDEX VALUE FOUND.

	DB	PBM	MWCS	MBCD	Partitions
ABC	0.5431	**2.7692**	**0.1107**	1.0080	5: [572, 423, 414, 290, 2]
CS	**0.1839**	2.7686	0.1108	1.0072	5: [571, 431, 407, 290, 2]
DE	0.5658	2.7681	0.1108	**1.0090**	5: [571, 421, 419, 288, 2]
GA	0.5665	2.4210	0.1235	0.9522	5: [571, 437, 401, 289, 3]
FA	0.6467	2.4180	0.1237	0.9555	5: [571, 422, 412, 293, 3]
PSO	0.6428	2.3860	0.1176	1.0010	5: [571, 397, 387, 344, 2]
SA	0.6024	2.3806	0.1310	0.9695	5: [571, 417, 412, 299, 2]
KM	0.5678	2.2220	0.0813	0.6145	4: [571, 426, 412, 292]

find only two possibilities. As a matter of fact, all metaheuristics ended with the same value $K_{opt} = 5$ in Table III. Besides, the number of elements of the obtained clusters are also more similar in the case of PBM (more on this issue later on).

From the point of view of the decision maker, the stability (represented here by the homogeneity of the results) of a certain method is essential to its credibility. Thus, the results reported in Table III seem to be more credible and, hence, deserved further investigation by the SCM experts. Anyway, in defense of the results in Table II, one can argue that a majority voting strategy could be adopted. In this case, the recommended optimal number of clusters would be $K_{opt} = 6$. But, it is worth mentioning that we are not only searching for this optimal number, but also if the clusters make sense from the point of view of the decision maker.

In this regard, if we take the three best-ranked algorithms suggesting $K_{opt} = 6$ in Table II, namely the CS, PSO and ABC algorithms, we can easily see that there are clusters with very few samples (e.g. 2 and 4 samples). These may simply be outliers. However, they could also be important customers that demand special attention, if their annual revenue per customer (x_2) is sufficiently high in comparison to the costs required to implement customer-specific logistical processes.

Proceeding with the analysis of the results in Table III, the use of the PBM as a fitness function led to more compact solutions with cluster centroids more separated away from each other, as revealed by the greater values of the MBCD index. In this regard, the top three solutions in this case obtained very similar PBM metrics. If we also take into consideration, the *a posteriori* value of the DB index, the CS algorithm achieved a much better DB index than the ABC algorithm. Thus, both ABC and CS solutions might be considered as the best ones in the scenario summarized in Table III.

Thus, in order to conclude our analyses, the issue of the distribution of elements (samples) among the clusters can be raised again. The suggested optimal number according to the solutions provided by the ABC and CS algorithms in Table III is $K_{opt} = 5$, but these solutions contain clusters with very few elements (2 elements, in this case). Do these very small clusters contain relevant data samples? Or, are they outliers? These clusters must be subjected to close scrutiny by the SCM experts in order to solve this issue, similar to the case described for CS, PSO and ABC for the DB Index results. A visual but a visual analysis may also be of help.

In Figure 2 we show the projections onto the first 3 principal components [28] of the data samples for each cluster solution reported in Table III. The colors were chosen to match the size order of the clusters, i.e., the largest cluster is in black, the second largest is in red, the third is in blue, and so on. We used this PCA-based cluster projection as a useful visualization tool to help us in the process of choosing the best cluster arrangements.

The projections of the three best-ranked metaheuristic algorithms in Table III seems to favor a solution with only $K_{opt} = 4$ clusters (represented by four colors: black, green, red and blue), a strong indication that those clusters with only two elements may not contain relevant information. Nevertheless, for the decision maker it is wise to analyze those clusters anyway.

As a final remark, one may argue that $K_{opt} = 4$ was originally the number suggested by the KM-based solution. However, the corresponding projections shown in Figure 2h reveal that the cluster segmentation provided by this algorithm is meaningless (i.e. only 3 clusters are relevant). Thus, the guidelines we are suggesting for the decision maker is to consider the cluster partitions provided by the ABC and CS metaheuristic optimization methods, as reported in Table III, with additional attention to be paid to the cluster with small number of samples (customers, in the context of the application of interest).

VII. CONCLUSIONS AND FURTHER WORK

In this paper we evaluated the performances of metaheuristic optimization algorithms for automatic clustering of a real-world supply chain data. The automatic clustering approach followed the methodology proposed in [1], in which the suitable number of clusters is a parameter to be found along the optimization process. A total of 7 metaheuristic algorithms were tested and their performances were contrasted. A comprehensive discussion of the achieved results was carried out in order to detect feasible clusters solution from the point of view of the supply chain managers.

To help in the decision for the best clustering solutions we compared the performances of the metaheuristic methods with that of standard (i.e. non-automatic) K-means algorithm. Furthermore, a PCA-based projection of the partitioned data samples for each clustering solution was carried out, indicating feasible data partitions. According to our analysis, the CS and ABC algorithms suing the PBM cluster validation index as objective function achieved the best overall performances.

Currently, we are testing multiobjective optimization strategies for automatic clustering of the dataset we worked in the current paper.

ACKNOWLEDGMENTS

The first two authors thank the financial support of FUNCAP, CNPq (grant no. 309451/2015-9) and NUTEC.

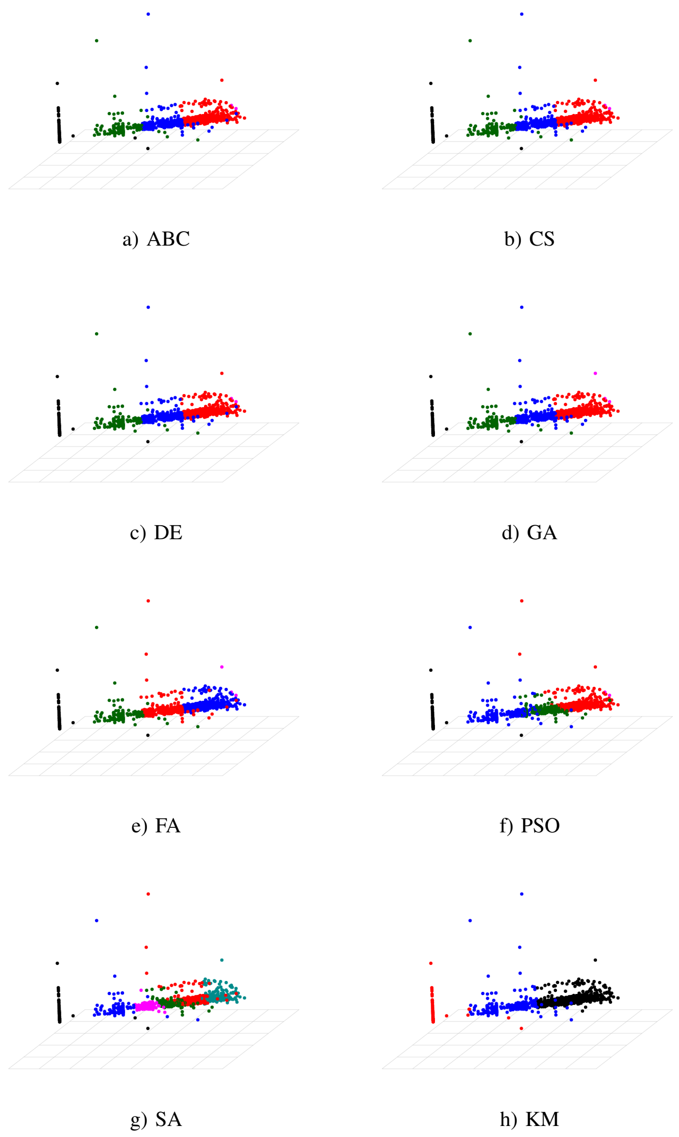

Fig. 2. Best clusterings obtained through optimization of the PBM index, ordered by best index value found.

References

[1] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man and Cybernetics-Part A*, vol. 38, no. 1, pp. 218–237, 2008.

[2] K. Butner, "The smarter supply chain of the future," *Strategy & Leadership*, vol. 38, no. 1, pp. 22–31, 2010.

[3] J. Griffiths, R. James, and J. Kempson, "Focusing customer demand through manufacturing supply chains by the use of customer focused cells: An appraisal," *International Journal of Production Economics*, vol. 65, no. 1, pp. 111–120, 2000.

[4] J. Collin, E. Eloranta, and J. Holmström, "How to design the right supply chains for your customers," *Supply Chain Management: An International Journal*, vol. 14, no. 6, pp. 411–417, 2009.

[5] U. Jüttner, M. Christopher, and J. Godsell, "A strategic framework for integrating marketing and supply chain strategies," *The International Journal of Logistics Management*, vol. 21, no. 1, pp. 104–126, 2010.

[6] M. Chattopadhyay, P. K. Dan, S. Majumdar, and P. S. Chakraborty, "Application of neural network in market segmentation: A review on recent trends," *Management Science Letters*, vol. 2, no. 2, pp. 425–438, 2012.

[7] S. Terlunen, G. A. Barreto, and B. Hellingrath, "Application and evaluation of multi-criteria clustering algorithms for customer-oriented supply chain segmentation," in *Logistics Management*, ser. Lecture Notes in Logistics, J. Dethloff, H.-D. Haasis, H. Kopfer, H. Kotzab, and J. Schoenberger, Eds. Springer, 2013, pp. 121–133.

[8] R. J. Kuo, Y. L. An, H. S. Wang, and W. J. Chung, "Integration of self-organizing feature maps neural network and genetic k-means algorithm for market segmentation," *Expert Systems with Applications*, vol. 30, no. 2, pp. 313–324, 2006.

[9] B. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster analysis*, 5th ed. Wiley, 2011.

[10] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[11] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "Validity index for crisp and fuzzy clusters," *Pattern Recognition*, vol. 37, no. 3, pp. 487–501, 2004.

[12] S. Das, A. Abraham, and A. Konar, *Metaheuristic clustering*. Springer, 2009.

[13] D. L. Davies, , and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, 1979.

[14] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[15] J. H. Holland, *Adaptation in natural and artificial systems*. MIT Press, 1975.

[16] S. Luke, *Essentials of metaheuristics*, 3rd ed. Lulu Raleigh, 2009.

[17] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[18] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[19] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[20] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-tr06, Erciyes University, Engineering fFaculty, Computer Engineering Department, Tech. Rep., 2005.

[21] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[22] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC'2009)*. IEEE Press, 2009, pp. 210–214.

[23] ——, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[24] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Advances in Swarm Intelligence*. Springer, 2010, pp. 355–364.

[25] Adexa, "The consumer packaged goods supply chain: optimized supply network management solutions," White Paper, 2013, Accessed in December 10, 2014. [Online]. Available: http://www.adexa.com/pdf/cpg_industry.pdf

[26] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016. [Online]. Available: http://www.R-project.org/

[27] B. Desgraupes, *clusterCrit: Clustering Indices*, 2016, r package version 1.2.7. [Online]. Available: http://CRAN.R-project.org/package=clusterCrit

[28] S. Haykin, *Neural networks and learning machines*, 3rd ed. Pearson, 2009.