# Decentralised Cooperative Agent-based Clustering in Intelligent Traffic Clouds

Jelena Fiosina, Maksims Fiosins and Jörg P. Müller *

Clausthal University of Technology,
Institute of Informatics,
Julius-Albert Str. 4, D-38678, Clausthal-Zellerfeld, Germany
{Jelena.Fiosina,Maksims.Fiosins}@gmail.com, Joerg.Mueller@tu-clausthal.de

**Abstract.** Contemporary traffic management systems will become more intelligent with advent of future Internet technologies. The systems are expected to become more simple, effective and comfortable for users, but this transformation will require the development of both new system architectures as well as enhanced processing and mining algorithms for large volumes of cloud data. In this study, we consider a conceptual architecture of a cloud-based traffic management system that applied to a multi-modal journey planning scenario. For this purpose, it is necessary to process large amounts of travel-time information. Information is collected by cloud service providers and processed for future route planning. In this paper, we focus on the data clustering step in the data mining process. The data collection and processing require an appropriate clustering algorithm to aggregate similar data. In particular, we support a process where a particular service provider can request additional information from others to be used in the clustering function, requiring a decentralised clustering algorithm. We present a cloud- based architecture for this scenario, develop a decentralised cooperative kernel-density based clustering algorithm, and evaluate the efficiency of the proposed approach using real-world traffic data from Hanover, Germany.

**Keywords:** Cloud computing architecture, decentralised data processing and mining, multi-agent systems, kernel density estimation, clustering

## 1 Introduction

Congested roads need to develop a new generation of Traffic Management Systems (TMS). These systems are very important for individual users, for example as drivers and pedestrians, business logistic operators and city public transport organizers. Such complex distributed systems can be well represented by multi-agent systems (MAS), which are based on multiple interacting and cooperating agents with intelligent behaviour.

*The research leading to these results has received funding from the EU 7th Framework Programme (FP7/2007-2013) under grant agreement No. PIEF-GA-2010-274881.

Future Internet capabilities such as cloud computing (CC) also influence TMS. CC aims at providing elastic services, high performance and scalable data storage to a large and ever increasing number of users [1]. CC systems provide large-scale infrastructure for high-performance computing and are dynamically adapted to user and application needs. Several common problems can be identified and several benefits obtained by the synergy between MAS and CC in an agent-based cloud computing (ABCC) paradigm. CC is mainly focused on the efficient use of computing infrastructure through reduced cost, service delivery, data storage, scalable virtualization techniques, and energy efficiency. In contrast, MAS are focused on the intelligent aspects of agent behaviour and their use in developing complex applications. In particular, CC can offer a very powerful, reliable, predictable and scalable computing infrastructure for the execution of MASs by implementing complex, agent-based applications for modelling and simulation. On the other hand, software agents can be used as basic components for implementing intelligence in clouds, making them more adaptive, flexible, and autonomic in resource management, service provisioning and large-scale application executions [14].

One of the key aspects of agent intelligence in ABCC is the capability to process and mine huge volumes of distributed data from various sources. Research activities have an inherent need to develop effective distributed data processing and mining algorithms for ABCC that take the needs and requirements of concrete traffic scenarios into account.

In this paper we consider travel time data processing that is distributed among cloud service providers. We focus on the problem of data clustering. Clustering is a descriptive data mining task used to partition a data-set into groups such that data objects in one group are similar to each other and are as different as possible from those in other groups. In cloud systems, data-sets are distributed among several providers, creating a need to develop distributed algorithms for data clustering. We develop a semi-parametric algorithm for distributed data clustering and consider some applications to ABCC-based intelligent TMS. We implement a kernel density (KD) technique based on contemporary computational statistics. It is a popular technique that reduces the search for clusters into a search for dense regions and allows finding arbitrary form clusters. This is accomplished using a so-called probability density function from which the given data set is assumed to have been generated [10].

The contribution of this study is the following: 1) a description of data processing stages in a cloud-based TMS that is applicable to a multi-modal journey planning scenario; 2) development of a novel decentralised cooperative agent-based KD clustering algorithm; 3) application of the proposed algorithm for the described scenario; 4) an experimental validation of the proposed clustering algorithm using real-world traffic data.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 formulates the considered TMS scenarios and presents the corresponding ABCC-based architecture. In Section 4, we present a KD clustering method. Section 5 suggests a decentralised, cooperative, agent-based KD

clustering algorithm and corresponding efficiency metrics. Section 6 reports the experimental results. The last section presents the conclusion and discusses the future work opportunities.

## 2 Related Work and State of the Art

By adapting to user and application needs, CC technologies, such as Future Internet and the Internet of Things (IoT), can enhance modern TMS and provide large-scale infrastructures for high-performance computing that are 'elastic' in nature [14]. CC technology is a worldwide priority research direction, and some authors (e.g., [12], [6]) have proposed work on next-generation TMS. Research areas are motivated and co-funded by private companies and municipalities in the areas of transport, logistics, communication and traffic management. Research in this area is still largely at the stage of scenario formulation and coordination protocols. One of the first cloud-based traffic network architecture, employing IoT components, has been proposed in [12].

Complex distributed systems are often modelled by MAS [7], [2]. Coupling CC with software agents opens new avenues for implementing intelligent cloud services. The convergence of interests between MAS, which require reliable distributed infrastructures, and CC systems, which needs intelligent software with dynamic, flexible, and autonomous behaviour, may result in new systems and applications [14]. Agents in such systems have the potential to be more intelligent and to possess special modules for intelligent data processing [10] and decision-making [7].

However, implementing a traffic cloud is far from easy. From an end user's point of view, the complexity of data and algorithms is hidden in the cloud. Users, ranging from traffic authorities to car drivers and automated components, expect to work with relatively simple web applications via mobile or embedded devices. These devices are permanently connected and can (theoretically) use all the information available from other users and system elements. A huge amount of available information must be found, collected, aggregated, processed and analysed for optimal decision-making and behaviour strategies. Such information is virtually centralized in cloud repositories, but should be managed physically in a decentralised fashion [6].

One of the key milestones in a path to more intelligent and up-to-date TMS is the continuous modification of existing methods and implementation of new modern technologies derived from the communication and data analysis fields. A decentralised regression forecasting model was considered in [5] and decision making methods in [7]. In this study, we focus on decentralised data clustering and the corresponding classification, which can be considered as a prerequisite step to implementation of forecasting and decision-making models.

In traffic research, the most popular clustering and classification problems are traffic state clustering [15] and participant behaviour clustering for group formation [11]. In this paper, we concentrate on clustering travel-time informa-

tion, which is part of the traffic state, trying to discover homogeneous traffic patterns that can be used with the common forecasting model.

KD clustering is a promising computational statistics tool that allows arbitrary shaped clusters to be discovered. Such non-parametric methods are well suited for exploring clusters, because no generative model of data is assumed. Instead, the probability density in the data space is directly estimated from data instances. Fast clustering, which is based on KD, was described by Hinnenburg and Gabriel [9]. The distributed (with a central authority) version of KD-based clustering (KDEC scheme) was considered in [10]. Another decentralised graph-oriented not KD clustering approach was presented in [13].

## 3 Cloud-based TMS architecture

We consider a future cloud-based TMS. The general architecture was proposed in [12] and then extended in [6]. The cloud-based TMS supposes the permanent connection of all participants and dynamically provides necessary services. The participants' applications that are run in the cloud are data-intensive, which necessitates the processing of large volumes of information to satisfy the participants' requests. In [6] we motivate the following data processing stages in a typical cloud-based TMS.

**Stage 1:** *Mining data from the IoT and its pre-processing.* All the participants of the cloud-based system have virtual representations as active IoT components (agents). The cloud system locates and collects the necessary data from different agents, and provides usual data mining operations (changes and outliers are detected, preliminary aggregation and dimensionality reduction are performed). The collected data are stored as historical information in the cloud and are used later as input data for ad-hoc network models (Stage 2).

**Stage 2:** *Ad-hoc network models.* The application-specific networks of virtual traffic participants are created, and the corresponding data models are used in order to estimate the important characteristics and parameters of these networks using the information collected in Stage 1 and for strategy optimization at Stage 3.

**Stage 3:** *Static decisions and initial strategy optimization.* Cloud applications use pre-calculated results of the ad-hoc network models from Stage 2 and the available historical information (including private information) about the traffic network to perform their pre-planning tasks. These models are also checked in the digital traffic network.

**Stage 4:** *Dynamic decisions and strategy update.* The pre-planned tasks from Stage 3 are executed, and updates are made according to the dynamic real-time situation extracted from the virtual agents. The aggregation of the pre-planned data and strategies with the dynamic ones is the most important problem at this stage.

We consider a dynamic multi-modal journey planning scenario [6]. In this scenario, the TMS helps travellers to plan and adjust a multi-modal, door-to-door journey in real-time. To provide recommendations to travellers, the cloud-

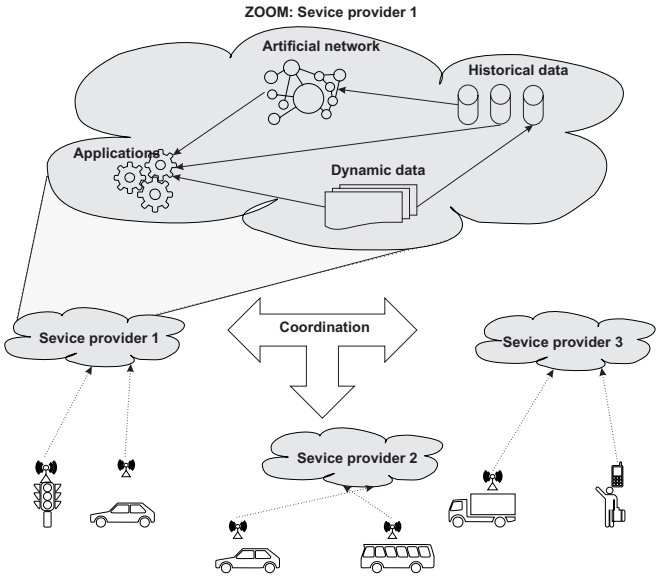


**Fig. 1.** System architecture

based TMS collects travel-time data in Stage 1 and organizes a historical travel information repository. In Stage 2 the travel times are continuously estimated for the travel maps. Stage 3 supposes construction of pre-defined routes for popular origin-destination pairs on the maps. In Stage 4 the actual information is taken into account and routes are updated correspondingly.

Note that there is no single 'central' cloud for this purpose. Instead, many *providers* may offer similar multi-modal journey planning services. They each collect information from subscribed travellers, and then create their own independent repositories, ad-hoc networks, and travel recommendations.

The providers are motivated by self-interest; their main goal is to maximize profit. To achieve this goal, they must balance two conflicting aspects of data processing. On the one hand, the providers are interested in maintaining a unique repository that provides clients with better services than those offered by competitors. On the other hand, the service providers are also interested in selling traveller information for profit (both to clients and to other providers).

After the creation of a (physical or virtual) data repository at Stage 1, each provider should process data in Stage 2 to prepare information for decisions in Stage 3. An important problem at Stage 2 is that of travel-time estimation for travel segments in the network. Different models, for example regression models, can be used for estimation [5]. However, to produce quality regression results, a single regression model should be used for each cluster, with potentially different models used across different clusters. Clustering of collected data is therefore one of the first tasks performed in Stage 2. In the next Section we describe a KD

technique of contemporary computational statistics, that allows finding clusters of arbitrary form [10], [9].

## 4 Kernel Density (KD) Clustering

Now let us formulate the clustering problem and describe the KD clustering algorithm. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in R^d$ be a dataset to be clustered into $k$ non-overlapping subsets $S_1, S_2, \ldots, S_k$.

Non-parametric clustering methods are well suited for exploring clusters of arbitrary form without building a generative model of the data. KD clustering consists of a two-step procedure: estimation and optimisation. During the estimation step, the probability density of the data space is directly estimated from data instances. During the optimisation step, a search is performed for densely populated regions in the estimated probability density function.

Let us formalize the estimation step. The density function is estimated by defining the density at any data object as being proportional to a weighted sum of all objects in the data-set, where the weights are defined by an appropriately chosen kernel function [10].

A KD estimator is

$$\hat{\Psi}^{[\mathbf{X}]}(\mathbf{x}) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}} |\mathbf{H}|^{-1} K\left(\mathbf{H}^{-1} \|\mathbf{x} - \mathbf{x}_i\|\right) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}}\left(\|\mathbf{x} - \mathbf{x}_i\|\right), \qquad (1)$$

where $\|\mathbf{x} - \mathbf{x}_i\|$ is a vector of coordinate distances between $\mathbf{x}_i$ and $\mathbf{x}$, $\mathbf{H}$ is a *bandwidth* matrix, $K(\mathbf{x})$ is a kernel function, $K_{\mathbf{H}}(\bullet) = |\mathbf{H}|^{-1} K\left(\mathbf{H}^{-1}\bullet\right)$ [8].

$K(\mathbf{x})$ is a real-valued, non-negative function on $R^d$ and has finite integral over $R^d$. We use the multivariate Gaussian function in our study: $K(\mathbf{x}) = (2\pi)^{-d/2} exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right)$. The bandwidth matrix $\mathbf{H}$ is a $d \times d$ positive-definite matrix that controls the influence of data objects and smoothness of the estimate. If no information is available with regard to correlation between factors, a diagonal matrix $\mathbf{H} = diag(h_1, \ldots, h_d)$ can be used.

Let us now formalize the optimisation step. This step detects maximum of KD and groups all of the data objects in their neighbourhood into corresponding clusters. We use a hill climbing method for KD maxima estimation with Gaussian kernels (DENCLUE2) [9] and modify the technique for the multivariate case. This method converges towards a local maximum and adjusts the step size automatically at no additional costs.

Each KD maximum can be considered as the centre of a point cluster. With centre-defined clusters, every local maximum of $\hat{\Psi}(\cdot)$ corresponds to a cluster that includes all data objects that can be connected to the maximum by a continuous, uphill path in the function of $\hat{\Psi}(\cdot)$. Such centre-defined clusters allows for arbitrary-shaped clusters to be detected, including non-linear clusters. An arbitrary-shape cluster is the union of centre-defined clusters that have maximum that can be connected by a continuous, uphill path.

The goal of the hill climbing procedure is to maximize the KD $\hat{\Psi}^{[\mathbf{X}]}(\mathbf{x})$. By setting the gradient $\nabla\hat{\Psi}^{[\mathbf{X}]}(\mathbf{x})$ of KD to zero and solving the equation $\nabla\hat{\Psi}^{[\mathbf{X}]}(\mathbf{x}) =$

0 for $\mathbf{x}$, we get:

$$\mathbf{x}^{(l+1)} = \frac{\sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}} \left( \left\| \mathbf{x}^{(l)} - \mathbf{x}_i \right\| \right) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}} \left( \left\| \mathbf{x}^{(l)} - \mathbf{x}_i \right\| \right)}. \tag{2}$$

The formula (2) can be interpreted as a normalized and weighted average of the data points. The weights for each data point depend on the influence of the corresponding kernels on $\mathbf{x}^{(l)}$. Hill climbing is initiated at each data point $\mathbf{x}_i \in \mathbf{X}$ and is iterated until the density does not change, i.e. $[\hat{\Psi}^{[X]}(\mathbf{x}_i^{(l)}) - \hat{\Psi}^{[X]}(\mathbf{x}_i^{(l-1)})]/\hat{\Psi}^{[X]}(\mathbf{x}_i^{(l)}) \leq \epsilon$, where $\epsilon$ is a small constant. The end point of the hill climbing algorithm is denoted by $\mathbf{x}_i^* = \mathbf{x}_i^{(l)}$, corresponding to a local maximum of KD.

Now we should determine a cluster for $\mathbf{x}_i$. Let $\mathbf{X}^c = \{\mathbf{x}_1^c, \mathbf{x}_2^c, \ldots\}$ be an ordered set of already identified cluster centres (initially, we suppose $\mathbf{X}^c = \emptyset$). First we find an index of the nearest cluster centre from $\mathbf{x}_i^*$ in the set $\mathbf{X}^c$:

$$nc(\mathbf{x}_i^*) = \arg\min_{j: \mathbf{x}_j^c \in \mathbf{X}^c} \left\| \mathbf{x}_j^c - \mathbf{x}_i^* \right\|.$$

If the nearest cluster centre is close to $\mathbf{x}_i^*$, then the point $\mathbf{x}_i$ is included in this cluster; otherwise, the point is used as a cluster centre to form a new cluster

$$\Lambda(\mathbf{x}_i) \leftarrow \begin{cases} nc(\mathbf{x}_i^*) & \text{if } \frac{\left\| \mathbf{x}^c_{nc(\mathbf{x}_i^*)} - \mathbf{x}_i^* \right\|}{\mathbf{x}_i^*} \leq \delta, \\ |\mathbf{X}^c| + 1 \text{ otherwise.} \end{cases}$$

where $\delta$ is a small constant and $\Lambda(x)$ is a class labeling function. In the second case, we also create a new cluster centre: $\mathbf{X}^c \leftarrow \mathbf{X}^c \cup \{\mathbf{x}_i^*\}$.

## 5 Decentralised KD-based clustering

In this section, we describe a cooperative method for sharing the clustering experience among agents in a network. While working with streaming data, one should take into account two main facts: (1) the nodes should coordinate their clustering experience over some previous sampling period, and (2) they must also adapt quickly to the changes in the streaming data without waiting for the next coordination action.

Let us first discuss the cooperation technique. Let $\mathbf{A} = \{A^j \mid 1 \leq j \leq p\}$ be a group of $p$ agents. Each agent $A^j \in \mathbf{A}$ has a local dataset $\mathbf{D}^j = \{\mathbf{x}_t^j \mid t = 1 \ldots, N^j\}$, $\mathbf{x}_t^j \in R^d$. To underscore the dependence of the KD function (1) from the local dataset of $A^j$, we denote the KD function by $\hat{\Psi}^{[\mathbf{D}^j]}(\mathbf{x})$.

Consider a case when some agent $A^i$ is unable to cluster a data point $\mathbf{x}_t^i$ in some future time moment $t$ because it does not have sufficient data nearby. By 'unable to cluster', we mean that this data point forms a new independent cluster after the optimisation step is performed. In this case, the agent $A^i$ sends a request to other neighbour agents by sending the data point $\mathbf{x}_t^i$ to them. Each agent $A^j$ that receives the request tries to classify the point $\mathbf{x}_t^i$ using its own

KD function $\hat{\Psi}^{[\mathbf{D}^j]}(\mathbf{x}_t^i)$ and performs an optimisation step to identify a cluster for this point. If the optimisation step is successful, meaning that this point belongs to an existing cluster, the agent replies to $A^i$ with information relevant to the neighbourhood of the requested point (parameters, artificial data points, random data points, etc.). Let us also define $\mathbf{G}^i \subset \mathbf{A}$, a group of agents that are able to reply to $A^i$ with clustering information.

Our model uses a two-phase protocol for performing communication between agents. First, since $A^i$ is unable to classify the data point $\mathbf{x}_t^i$, the information is sent to other agents. In response to the help-request, the neighbours $A^j$ send parameters from their estimated KD functions. Since the KD function is non-parametric and estimated directly from observations, we approximate the function with a mixture of multi-dimensional Gaussian distributions. Agent $A^j$ identifies cluster associated with point $\mathbf{x}_t^i$ and performs the approximation of clusters with a mixture of normal distributions. Next, $A^j$ transmits the cluster parameters (weight, mean and covariance matrix). The agent $A^i$ adds this information to its KD and updates its clusters. Since parameter transmission requires less data, this approach requires less transmission, however, the approximation reduces the cluster shapes to a union of ellipsoids.

Let us consider an approximation step that approximates KD functions with a mixture of multivariate normal distributions. This step can be achieved with the expectation maximisation (EM) algorithm proposed by Dempster [4]. The approach is widely used for calculation of the maximum likelihood estimate of mixture models.

In a mixture model, the probability density function is

$$f(\mathbf{x}; \mathbf{\Theta}) = \sum_{b=1}^{B} \pi_b f_b(\mathbf{x}; \Theta_b), \tag{3}$$

where $\pi_b$ are positive mixing weights that sum to one, $f_b$ are component density functions parameterized by $\Theta_b$, and $\Theta = \{\pi_b, \Theta_b\}$ are the model parameters. Each observation is assumed to be from one of the $B$ components. A common choice for component density is a multivariate normal distribution with parameters $\Theta_b = (\mu_b, \mathbf{\Sigma}_b)$, where $\mu_b$ is a mean and $\mathbf{\Sigma}_b$ is a covariance matrix. Given a set of independent observations $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^v\}$ in $\mathbf{x}^j \in \mathbf{R}^d$, the objective is to fit such a model to the data.

In the EM procedure, the expected likelihood of a given data-set is iteratively maximized. Let $\mathbf{z}^i \in \{0,1\}^B$ be the membership indicator variable such that $z_b^i = 1$ if $\mathbf{x}^i$ is generated by $f_b(\cdot)$ and 0 otherwise.

The E step simplifies to computing the conditional probabilities

$$\left\langle z_b^i \right\rangle = P\left\{ z_b^i = 1 | \mathbf{x}^i; \Theta^{old} \right\} = \frac{\pi_b f_b(\mathbf{x}^i; \Theta^{old})}{\sum_l \pi_l f_l(\mathbf{x}^i; \Theta^{old})}. \tag{4}$$

In the M step, we have an update rule in closed form:

$$\hat{\pi}_b = \frac{1}{v} \sum_i \left\langle z_b^i \right\rangle, \; \hat{\mu}_b = \frac{\sum_i \left\langle z_b^i \right\rangle \mathbf{x}^i}{\sum_i \left\langle z_b^i \right\rangle}, \tag{5}$$

$$\hat{\Sigma}_b = \frac{\sum_i \left\langle z_b^i \right\rangle (\mathbf{x}^i - \hat{\mu}_b)(\mathbf{x}^i - \hat{\mu}_b)^T}{\sum_i \left\langle z_b^i \right\rangle}. \tag{6}$$

The algorithm alternates between the E and the M steps until convergence is achieved.

We assume that each helping-agent $A^j \in \mathbf{G}^i$ receives data point $\mathbf{x}_t^i$ and tries to classify it. If it is successful, $A^j$ determines that $\mathbf{x}_t^i$ belongs to a specific cluster and executes the EM-algorithm with this cluster. This algorithm approximates the cluster using a mixture of $B^j$ multidimensional normal distributions with parameters $\Theta^j = \{\mu^j, \Sigma^j, \pi^j\}$, where $\mu^j = (\mu_1^j, \ldots, \mu_{B^j}^j)$, $\Sigma^j = (\Sigma_1^j, \ldots, \Sigma_{B^j}^j)$ and $\pi^j = (\pi_1^j, \ldots, \pi_{B^j}^j)$, which are then returned to $A^i$.

After receiving all answers, the agent $A^i$ has a vector of the parameters $\{\Theta^j\}$. The answers $\{\Theta^j\}$ can be interpreted by the agent $A^i$ as data points $\mu^j$ with the only difference being that the additional weights $\pi^j$ and bandwidths from $\Sigma^j$ should now be taken into account. Denote $\hat{\mathbf{D}}^i$ as a dataset of the agent $A^i$ that includes the received answers. Density estimates (1) of each agent are additive, i.e. the aggregated density estimate $\hat{\Psi}^{[\hat{\mathbf{D}}^i]}(\mathbf{x})$ can be decomposed into a sum of local density estimates and answers:

$$\hat{\Psi}^{[\hat{\mathbf{D}}^i]}(\mathbf{x}) = w_i \hat{\Psi}^{[\mathbf{D}^i]}(\mathbf{x}) + \frac{1 - w_i}{\sum\limits_{\substack{A^j \in G^i \\ b \in B^j}} \pi_b^j \cdot \sum\limits_{A^j \in G^i} B^j} \sum_{\substack{A^j \in G^i \\ b \in B^j}} \pi_b^j K_{\Sigma_b^j}(\|\mathbf{x} - \mu_b^j\|), \tag{7}$$

where $w_i$ is the weight assigned to own observations of the agent.

To measure the **clustering similarity** [3] among the agents $A^i \in \mathbf{A}$ we use the following representation of a class labeling by a matrix $C$ with components:

$$C_{i,j} = \begin{cases} 1 \text{ if } x_i \text{ and } x_j \text{ belong to the same cluster and } i \neq j, \\ 0 \text{ otherwise.} \end{cases}$$

Let two labelings have matrix representations $C^{(1)}$ and $C^{(2)}$, respectively. We define a dot product that computes the number of pairs clustered together $\left\langle C^{(1)}, C^{(2)} \right\rangle = \sum_i \sum_j C_{i,j}^{(1)} C_{i,j}^{(2)}$. The Jaccard's similarity measure can be expressed as

$$J(C^{(1)}, C^{(2)}) = \frac{\left\langle C^{(1)}, C^{(2)} \right\rangle}{\left\langle C^{(1)}, C^{(1)} \right\rangle + \left\langle C^{(2)}, C^{(2)} \right\rangle - \left\langle C^{(1)}, C^{(2)} \right\rangle}. \tag{8}$$

## 6 Experimental simulation results and case studies

We simulated a traffic network in the southern part of Hanover (Germany). The network contained three parallel and five perpendicular streets, creating fifteen intersections with a flow of approximately 5000 vehicles per hour.

The service providers solved a travel time clustering problem using the factors listed in Table 1. They received information about the centrally estimated system

variables (such as average speed, number of stops, congestion level, etc.) for this city district from TMS, combined it with their historical information, and made adjustments according to the information of other participants.They used the autonomous KD clustering (Section 4) and implemented the decentralised cooperative clustering algorithm (Section 5).

**Table 1.** System factors influencing the travel time

| Variable | Description |
|---|---|
| $Y$ | travel time (min); |
| $X_1$ | length of the route (km); |
| $X_2$ | average speed in the system (km/h); |
| $X_3$ | average number of stops in the system (units/min); |
| $X_4$ | congestion level of the flow (veh/h); |
| $X_5$ | number of traffic lights in the route (units); |
| $X_6$ | number of left turns in the route (units). |

Note that the clustering of the factors $X_1 - X_6$ is performed together with the dependent variable $Y$. This clustering step can be considered as a pre-processing of initial data for the future forecasting with regression models. Different travel time $Y$ forecasting models were obtained inside different clusters, that allows better fit of the models and better prediction considered in [5].

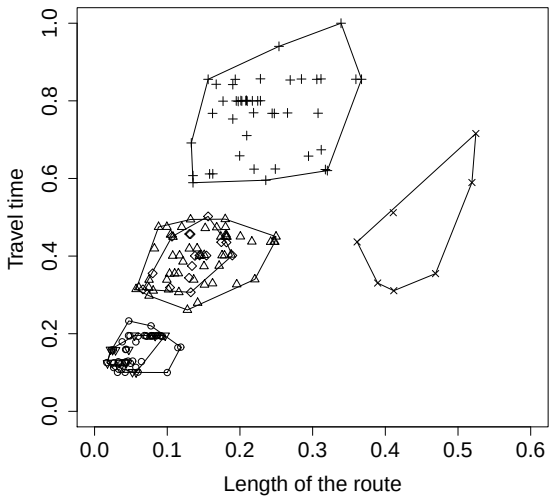


**Fig. 2.** A $X_1 - Y$ projection of 200 observations

We simulated 10 agents with initial experience varied in range from 10 to 200 observations. Most simulation experiments ran for 200 time units. We assumed that all observations were homogeneous and the agents tried to estimate the same clusters. We normalized the initial data before the simulation execution.

To provide a visual overview of data, we presented the projection of 200 observations to the $X_1 - Y$ plane with the corresponding six clusters (Fig. 2). The visual intersection of the clusters in this projection is due to the difference of the point values in other dimensions.

For more accurate clustering, the agents used the presented decentralised cooperative KD clustering algorithm. Cooperation among the agents allowed improving clustering quality.

Let us illustrate one data synchronization step presented at Fig. 3. The requesting agent asks for help for point A (top left). The helping agent clustered the point using its own data, detected the corresponding cluster (bottom left), approximated it with the mixture of three normal distributions (shown as ellipses for two dimensional case with centres in B, C, D) and sent the corresponding parameters to the helping agent (bottom right). The helping agent added the obtained parameters as data points to its data and made new clustering (top right). This allowed to improve clustering similarity of these two agents from 0.011 to 0.037 as well as clustering similarity of the requesting agent with an 'ideal' clustering from 0.004 to 0.006.

A system dynamics for a different number of transmitted points is shown at Fig. 4. Clustering similarity (right) increased faster for a bigger number of the estimated and transmitted components of normal distributions $B^j$ for cluster approximations, but the number of communication events (left) decreased faster. Note, however, that one communication event was more expensive for a bigger number of transmitted points, but supplied more information.

Quality of the agent models was also checked by a cross-validation technique (Fig. 5) at the beginning (left) and at the end (right) of the simulation. These histograms shown a probability distribution of a similarity, which peak moved to bigger value after the coordination procedure.

## 7 Future Work and Conclusions

We developed the decentralised coordinated KD clustering approach for agent-based cloud computing architecture of intelligent transport system. The data coordination scheme is based on the transmission of parameters of multidimensional normal distribution, which approximate the cluster to which the requested point belongs. An experimental validation of the developed algorithm was also performed. Our future work is devoted to the development of new coordination schemes in proposed decentralised clustering approach.

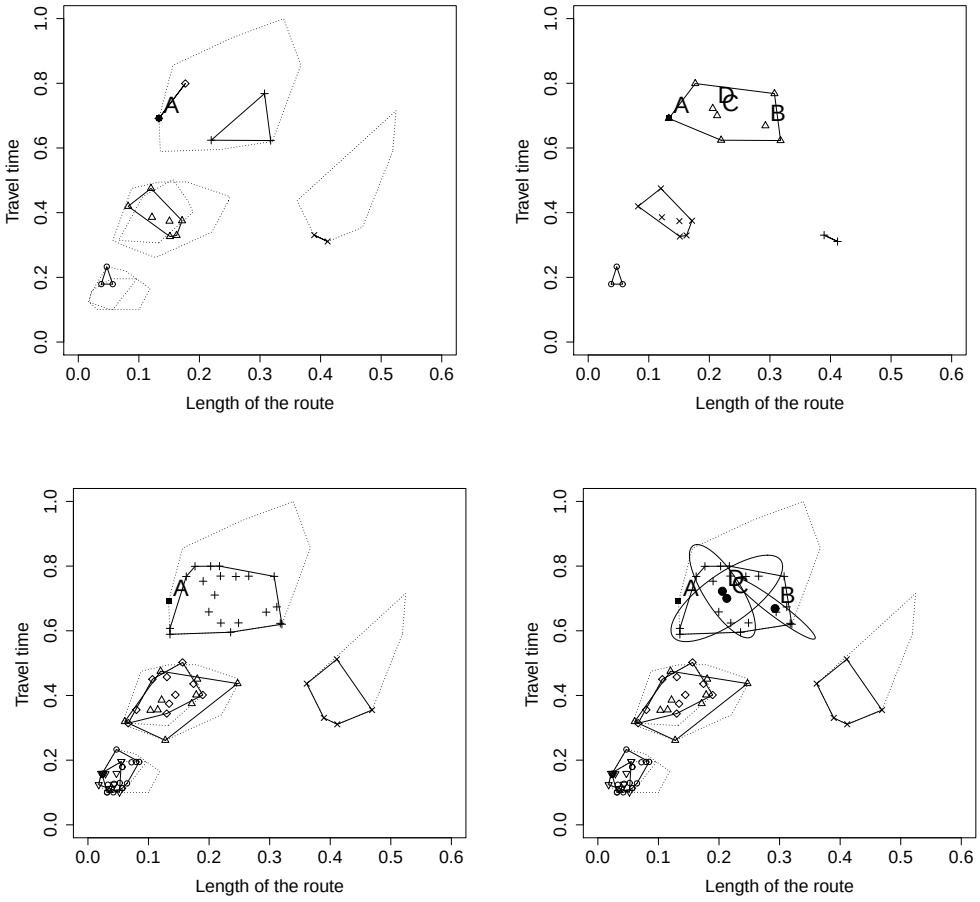


**Fig. 3.** A communication step between the requesting (top) and helping (bottom) agents.

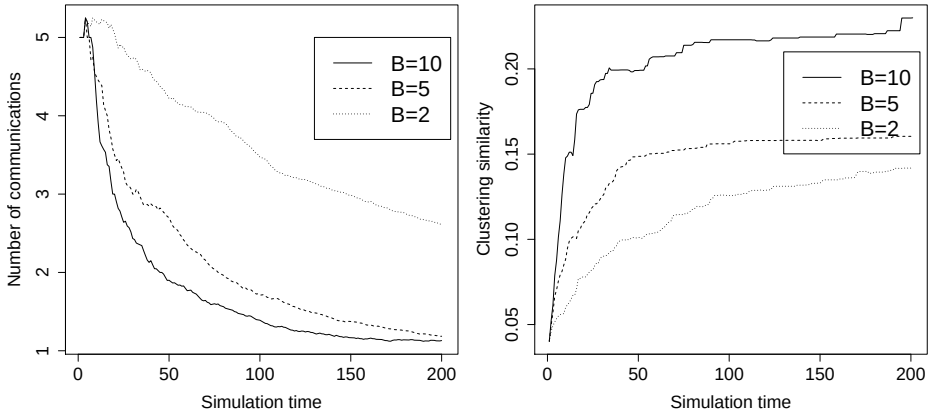


**Fig. 4.** A number of communication events (left) and similarity of agents' clusters (right) over time depending on $B$ components in a mixture of multidimensional normal distributions

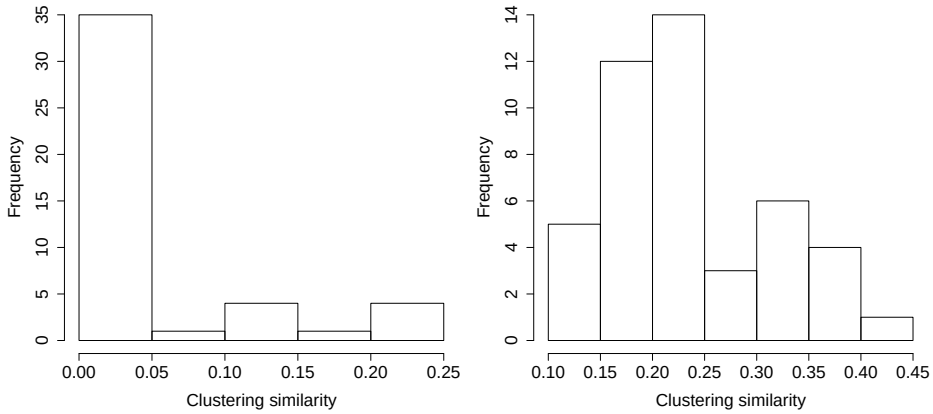


**Fig. 5.** Frequencies of clustering similarity for each pair of agents at the beginning (left) and at the end (right) of simulation

## References


1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Communications of the ACM **53**(4), 50–58 (2010)
2. Bazzan, A.L.C., Klgl, F.: A review on agent-based technology for traffic and transportation. The Knowledge Engineering Review **FirstView**, 1–29 (2013)
3. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Pacific Sym. on Biocomputing 7, pp. 6–17 (2002)
4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. of the Royal Stat. Society. Series B **39**, 1–38 (1977)
5. Fiosina, J., Fiosins, M.: Chapter 1: Cooperative regression-based forecasting in distributed traffic networks. In: Q.A. Memon (ed.) Distributed Network Intelligence, Security and Applications, pp. 3–37. CRC Press, Taylor and Francis Group (2013)
6. Fiosina, J., Fiosins, M., Müller, J.P.: Mining the traffic cloud: Data analysis and optimization strategies for cloud-based cooperative mobility management. In: Proc. of Int. Sym. on Management Int. Systems, *Adv. in Int. Syst. and Comp.*, vol. 220, pp. 25–32. Springer-Verlag, Berlin Heidelberg (2013)
7. Fiosins, M., Fiosina, J., Müller, J.P., Görmer, J.: Reconciling strategic and tactical decision making in agent-oriented simulation of vehicles in urban traffic. In: 4th Int. ICST Conf. on Simulation Tools and Techniques (SimuTools'2011) (2011)
8. Härdle, W., Müller, M., Sperlich, S., Werwatz, A.: Nonparametric and Semiparametric Models. Springer-Verlag, Berlin Heidelberg (2004)
9. Hinneburg, A., Gabriel, H.H.: DENCLUE 2.0: Fast clustering based on kernel density estimation. In: Proc. of IDA'07, Adv. in Intelligent Data Analysis VII, *LNCS*, vol. 4723, pp. 70–80. Springer-Verlag, Berlin Heidelberg (2007)
10. Klusch, M., Lodi, S., Moro, G.: Agent-based distributed data mining: The KDEC scheme. In: AgentLink, pp. 104–122 (2003)
11. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: A partition-and-group framework. In: ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'07), pp. 593–604. Beijing (2007)
12. Li, Z., Chen, C., Wang, K.: Cloud computing for agent-based urban transportation systems. IEEE Int. Systems, IEEE Computer Society **26**(1), 73–79 (2011)
13. Ogston, E., Overeinder, B., van Steen, M., Brazier, F.: A method for decentralized clustering in large multi-agent systems. In: Proc. of 2nd Int. Conf. on Autonomous Agents and Multiagent Systems, pp. 789–796 (2003)
14. Talia, D.: Cloud computing and software agents: Towards cloud intelligent services. Proc. of the 12th Workshop on Objects and Agents **741**, 2–6 (2011)
15. Weijermars, W., van Berkum, E.: Analyzing highway flow patterns using cluster analysis. In: Proc. of the 8th Int. IEEE Conf. on ITS, pp. 831–836. Vienna (2005)