Mobile Netw Appl (2013) 18:766–787
DOI 10.1007/s11036-013-0468-5

A Cluster-Based Beaconing Approach in VANETs: Near Optimal Topology Via Proximity Information

Yair Allouche · Michael Segal

Published online: 2 October 2013

Abstract A key component for safety applications in Vehicular ad-hoc network (VANET) is the use of periodic beacon messages which provide vehicles with a real-time vehicle proximity map of their surroundings. Based on this map, safety applications can be used for accident prevention by informing drivers about evolving hazardous situations. In order to allow synchronized and cooperative reactions, the target of this work is to design a beacon dissemination process that provides a real-time, broad and coordinated map under the challenging VANET conditions. In order to obtain the desired map, we consider an aggregation-dissemination based scheme for a beacon dissemination process that based on top of a cluster-based topology. To this end, we propose the Distributed Construct Underlying Topology (D-CUT) algorithm tailed specifically to provide an optimized topology for such beacon dissemination process. To deal with the heavy load of beacon messages required for an accurate and broad map, we propose a topology that allows the execution of extensive but reliable spatial bandwidth reuse. Our D-CUT algorithm exploits the real-time and coordinated map for constructing an adaptive and robust topology to deal with the dynamic nature of the VANET environment. We present theoretically provable bounds demonstrating the ability of the algorithm to deal with the dynamic nature of the VANET environment supported by simulation results.

Keywords Beacon dissemination · Distributed algorithm · Optimal clustering assignment · Self-organizing topology

1 Introduction

Vehicular ad-hoc network (VANET) is a promising branch of traditional MANET. VANET is designed to provide wireless communication between vehicles and between vehicles and nearby roadside equipment. This communication intends to improve both safety and comfort on the road. VANET has a number of difficulties regarding the traditional MANET. Due to the dynamic nature of VANET environments, configuration is always changing, where links may appear and disappear very quickly and vehicle density is constantly changing. On the other hand, VANET has some inherent advantages over the traditional MANET. It is generally assumed that vehicles will be aware of their own geographical position. In addition, vehicles in a VANET environment move in an organized fashion within the constraints of traffic flow.

A key component in safety applications are the periodic beacon messages which provide vehicles with a broad and accurate *vehicle proximity map* of their surroundings. Based on this map, safety applications—usually referred to as Cooperative Awareness applications—can be used for accident prevention by informing drivers about *evolving* hazardous situations. In addition, an accurate vehicle proximity map can facilitate other essential multi-layer objectives such as optimized geographic oriented forwarding [1] and addressing methodologies. From a routing point of view, high levels of awareness can be very beneficial in terms of route discovery, end-to-end delay, and number of retransmissions [2]. Torrent-Moreno et al. [3] propose a transmit power control method, based on the vehicles' location proximity, to control the load of beacon messages.

In order to be used as a reliable infrastructure for safety applications, the surrounding vehicle proximity map should be as accurate as possible. Hence, while considering a fully deployed high-density vehicular scenario combined with the dynamic topology of the vehicular environment (e.g., a free highway), creating a broad and accurate vehicle proximity map becomes challenging. Such an accurate estimation in a dynamic environment requires a high transmission frequency of beacon messages, in broadcast fashion, from numerous nearby vehicles; which, in turn, results in a high data load on the channel. Thus, beacon dissemination methodology is measured according to its ability to provide an accurate map under such a high load on the channel.

Y. Allouche · M. Segal (✉)
Communication Systems Engineering Department, Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: segal@cse.bgu.ac.il

In this paper we suggest a self-organizing cluster-based topology to serve as the infrastructure for a beacon dissemination process. This process is designed to replace the traditional multipoint-to-multipoint transmission of beacon messages by a cluster-based aggregation-dissemination process. For this purpose, the network is partitioned into clusters of adjacent vehicles (see Fig. 1). Each cluster contains a designated vehicle referred to as the *clusterhead,* connected by one-hop *intra-cluster links* to its cluster members. The second level of the topology consists of multi-hop, *inter-cluster links* that connect adjacent clusterheads.

On top of this topology, we consider the following three-phase beacon dissemination process. In the first phase, beacons in the same cluster are aggregated by clusterheads. In the second phase, clusterheads disseminate the compressed aggregated beacon to their adjacent clusters. In the final phase, clusterheads broadcast the aggregated information to all their cluster members, providing each vehicle with a local vehicle proximity map. As the map is disseminated from a single source and in one broadcast transmission, each cluster member successfully receiving this broadcast transmission receives the same vehicle proximity map as its surroundings. Later we will show how this coordination is exploited by our algorithm.

Into the aforementioned process, we suggest integrating contention-free medium access control (MAC) protocols. This notion has been suggested in the past [4, 5] because of the following twofold benefits: First, intra-cluster channel access synchronization provides contention-free access between cluster members. Second, bandwidth efficiency is achieved by bandwidth reuse among clusters. However, to date, this bandwidth reuse has not yet been utilized properly due to interference from adjacent clusters.

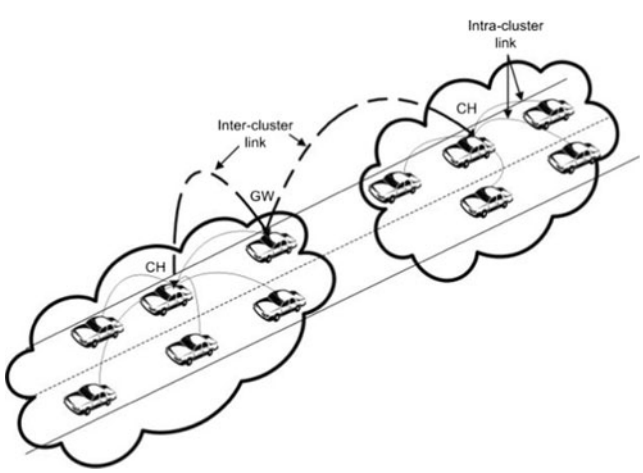

Fig. 1 The hierarchical network topology described in this paper created by grouping sets of sequential vehicles into clusters. At the intra-cluster level, the members of each cluster are linked to a designated clusterhead (CH). At the inter-cluster level, CHs are linked, if needed, via gateways (GWs), to their adjacent clusters

The target of this work is to design a clustering scheme that provides an optimized topology for an efficient and reliable beacon dissemination process. To provide this, we want a process that executes extensive but reliable spatial bandwidth reuse. To this end, the inter-cluster interference must be addressed. This work aims to reduce this interference by geographically optimizing the topology, and in this way, creating reliable bandwidth reuse. So to geographically optimize the topology, we require that the clusters be as dense and as far apart from each other as possible. Furthermore, we require limiting cluster size in order to guarantee that each vehicle has contention-free channel access on which to send its message. Once limited, our objective is to increase the cluster to its maximal size; thereby allowing the most efficient utilization of the allocated bandwidth. Later, we define an optimization problem according to the above objectives.

Requiring that the clusters be as far apart from each other as possible comes at the expense of less-efficient inter-cluster links. However, since the beacon dissemination process is measured by its ability to cope with a high data load carried by intra-cluster links, reliable spatial bandwidth reuse is of the essence.

In this paper we introduce the Distributed Construct Underlying Topology (D-CUT) algorithm, which aims to provide the desired topology under the challenging VANET conditions. On top of this topology, our beacon dissemination process provides each vehicle a real-time and coordinate vehicle proximity map to be used by safety applications. In order to cope with the challenging VANET conditions, the algorithm uses this available real-time and coordinated map as the building block for maintaining the topology. By using the coordinated map as its input, the D-CUT algorithm creates a synergetic relationship with the beacon dissemination process. On one hand, the more coordinates and the greater the accuracy of the vehicle proximity map, the better the algorithm's performance. On the other hand, better algorithm performance leads to a better beacon dissemination process, which results in a more accurate and coordinated proximity map.

In more detail, for a clustering strategy to be feasible for VANET, it must promptly react to the highly dynamic behaviors of vehicular networks. To obtain adaptivity, the D-CUT algorithm logically partitions the local vehicle proximity map into road sections where each section contains geographically optimized clusters. Given the real-time location of the nearby vehicles, the algorithm updates the partitioning according to the most recent topological changes while aiming to maintain geographically optimized clusters. The algorithm coordinates its operation by exploiting the coordinated map. Later, we will present theoretical and simulation studies to demonstrate the ability of the D-CUT algorithm to self-start and maintain the geographically optimized clusters under the dynamic nature of VANET environments.

Though hierarchical topology has many advantages, the downside is its over-sensitivity to clusterhead failures. In order

to ease this effect, the algorithm grants clusterheads a temporary, easy-replaceable leadership position rather than a stable one. So instead of following the common approach, which begins with clusterhead selection and the consequent formation of a cluster, this algorithm reverses the process by starting with cluster formation and only then chooses the temporary clusterhead. To deal with the frequent clusterhead replacement, the algorithm provides a straightforward and robust clusterhead election procedure that is based on the available coordinated vehicle proximity map.

This paper is organized as follows: In Section 2, we summarize other approaches for building a hierarchical topology in VANET. In Section 3 we give a formal definition of the clustering optimization problem considered in this paper. Then, in Section 4, we describe the D-CUT algorithm and in Section 5, we show theoretically provable bounds for the algorithm's performance. In Section 6, we show a simulation study that supports our analytical results. This paper concludes with Section 7.

2 Related work

There has been extensive research on the multi-layered benefits of cluster-based protocols in Vehicular Ad Hoc Networks. The focus has been on developing cluster-based MAC protocols, as in [4, 5]; cluster-based routing protocols, as in [6, 7]; cluster-based broadcasting protocols, as in [8, 9]; and security protocols, as in [10]. Generally, in cluster-based MAC protocols clusterheads are responsible for management tasks regarding the medium access. In cluster-based routing protocols, efficiency is achieved by flooding the routing control message on top of the topology backbone. In order to reduce the redundant retransmissions known as the "broadcast storm problem" in cluster-based broadcasting protocols, clusterheads and some selected gateway vehicles are given sole responsibility for rebroadcasting. From a security perspective, cluster-based data aggregation can contribute to better data correctness by crosschecking for consistency verification of the data aggregated from the cluster members.

The prevailing clustering formation strategy (e.g., [4, 5, 11]) in VANET is to distribute the state of vehicles—commonly: undecided, member, gateway, or clusterhead—on the regular transmission of beacons. Each vehicle chooses its appropriate state, according to the state of the vehicles nearby. An undecided vehicle will join the first clusterhead from which it hears a beacon, and if the vehicle does not hear from a clusterhead within a given time period, it will become a clusterhead itself. When two clusterheads come within a predefined range, a clusterhead election procedure is applied in order to guarantee a minimal range between adjacent clusterheads. This range can be the actual spatial distance, as in [12]. The implication is that a clusterhead is required to hold real-time knowl`edge of the adjacent clusterhead's position. To avoid this requirement, the range can be estimated according to the received signal strength [5]. In [4], when vehicles receive a beacon message from more than one clusterhead, it changes its state to gateway. However, in [13], Kenichi et al. show that if no special criteria are used, under this strategy almost all non-clusterhead vehicles change their state to gateways, increasing the number of vehicles involving in packet relaying, as well as duplicated packets and the probability of packet collision. To reduce the number of gateways involved in the packet relaying, in [14] Kayis et al. propose to select the optimal gateway in terms of minimizing the speed difference between gateways and the corresponding clusterhead vehicle.

Broadly speaking, the above clustering schemes can be sub-partitioned according to the objectives of the clustering scheme, which it takes into account during the clusterhead election procedure. In order to reduce the cluster reorganization overhead, a widespread objective is to try to maintain a stable clustering [4, 8, 9, 12, 15]. One approach to obtain stability (see for example [4, 12]) is to assign higher priority in the clusterhead election procedure for the vehicle with small speed deviation from the surrounding vehicles' average speed. In [12], Wang et al. suggest taking into account the trip duration in the election procedure. A vehicle that is about to travel for a longer time is assigned higher priority. Another approach to increase stability is to choose the vehicle with the longest clusterhead duration at the first clusterhead election procedure [9]. Two additional objectives relevant to our study are controlling cluster size and producing non-overlapping clusters. The common approach for controlling cluster size is by setting a predefined maximum distance between a clusterhead and its members. However, this approach does not scale well and will poorly adapt to the diverse and constantly changing density introduced in the vehicular environment. In [11], Fan et al. suggest using a common fixed upper bound on all cluster sizes. The implication is that the clusterhead may reject vehicles within range from joining the cluster due to resource exhaustion. Consequently, the lower bound for distance between adjacent clusterheads is not assured. In order to produce non-overlapping clusters, Wang et al. [12] suggest electing a clusterhead that has the highest priority in its one-hop neighborhood and the highest priority in the one-hop neighborhood of one of its one-hop neighbors. However, this strategy assures non-overlapping clusters only at the time of the clusterhead election procedure, which in the dynamic vehicular environment may last for a short period of time.

As discussed above, in order to increase robustness we look for a clustering strategy that starts with group formation and only then chooses the temporary clusterhead. The second approach for clustering formation is based on this idea (e.g., [10, 13]). Groups are defined by dissecting roads into predetermined area cells. This is obviously a very simple

and efficient approach as each vehicle will automatically know to which cluster it belongs and the clusterhead will be automatically chosen by its proximity to the cell's center. However, it may create unstable groups, so even when groups of vehicles are traveling together at the same speed, the group will constantly re-divide. Also this method is non-scalable, as the cell sizes are predetermined and cannot be adapted to different traffic densities. In this work, we suggest dissecting the road in a dynamic manner rather than in a static one. The benefit is that it will result in high adaptivity, allowing us to set objectives that insure the quality of our clustering.

Since used by life-critical safety applications, security is a fundamental aspect of any beaconing process. Rather than reinventing the wheel, we refer readers to [15], at which the authors present a secure beaconing process at which beacon messages are digitally signed and carry a certificate to confirm valid network participants.

3 System model and problem definition

In this work we consider clustering scheme in a multi-lane highway scenario. We suggest leveraging the organized movement of vehicles on road systems in the following manner. The road is divided at each road intersection into road segments with one entrance and one exit. Each such segment will be clustered independently. The reason for this is that in this work we consider a cluster-based topology for inter-vehicle communication. Since Road Side Units (RSUs) are expected to be deployed on road intersections in order to facilitate safety applications such as Blind Merge Warning,[1] and as most of the communication will be from vehicle to RSU and back, we assume that RSUs will act as clusterheads for a predefined area around them. The exact manner in which RSUs integrate into the hierarchical topology is outside the scope of this work.

Moreover, in our clustering scheme only groups of vehicles traveling in the same direction are clustered. This is viable as vehicles traveling in the same direction share similar moving patterns due to traffic laws and road structures, thereby creating a stable topology. In addition, when considering highway or suburban roads, vehicles traveling in opposite directions are commonly separated by traffic barriers and, therefore, their beacon messages are less relevant.

In what follows we will describe the geographic clustering optimization problem for the above scenario, derived from efficient and reliable beacon dissemination objectives. First we describe the objectives at the cluster level. Based on those objectives we define the criteria for a valid solution to our clustering problem. Then, we describe the objectives at the performance topology level, and based on them we define our optimization problem.

Before diving into a detailed description of the problem definition, some notations and definitions are required (see Fig. 2). We are given a network N with n ordered nodes $U=\{u_1,u_2,\ldots,u_n\}$ that are moving along a road from left to right (we will discuss this assumption later). Instead of denoting the location of nodes explicitly, we use their relative locations. Let us denote by $D=\{d_0,d_1,\ldots,d_n\}$ the set of inter-distances such that d_i is the inter-distance between u_i and u_{i+1}. The inter-distances d_0,d_n denote the space at the edge of the model and are set to ∞. In some cases, we will need to observe subsets of the sets U and D. Hence, let $U(d_i,d_j)$ be the subset of U framed by the inter-distances d_i,d_j, i.e., $U(d_i,d_j)=\{u_{i+1},u_{i+2},\ldots,u_j\}$. Similarly, let $D(d_i,d_j)$ be the D subset $\{d_{i+1},d_{i+2},\ldots,d_{j-1}\}$. To indicate that one or both of the endpoints is to be included in the set, we substitute a square bracket for the corresponding parenthesis, e.g., $D[d_i,d_j)=\{d_i,d_{i+1},\ldots,d_{j-1}\}$. In addition, let us denote by $S=\{C_1,C_2,\ldots,C_m\}$ the set of clusters such that C_i is a set of consecutive nodes that forms the i'th cluster in the set, and m is the number of clusters in the model. Accordingly, let $G=\{g_0,g_1,\ldots,g_m\}$ be the set of inter-cluster gaps such that g_i represents the inter-distance located between the clusters C_i and C_{i+1}, and g_0,g_m represent the end-points d_0,d_n, respectively. Notice that according to the above notations $C_i=U(g_{i-1},g_i)$. In some cases, we will want to refer to the set of inter-cluster gaps G at a specific D-CUT iteration. For this purpose, let $G(t)$ be the set of inter-cluster gaps at iteration t.

Remark The D-CUT algorithm is based on comparing the length of inter-distances and gaps. In order to deal with ties in gap or inter-distance comparisons, the gap/inter-distance having the smaller index wins.

At the cluster level, we look for star topology, which allows one hop aggregation/dissemination. This objective requires the existence of at least one clusterhead candidate that covers the entire cluster population within its transmission range. However, to increase the robustness of the topology we require the existence of at least $p_{min}>1$ clusterhead candidates. To ensure a valid solution for any possible configuration, we demand all cluster members to be clusterhead candidates when the size of the cluster is at most p_{min}. Our second objective is to limit the cluster size in order to allocate to each cluster member an orthogonal channel resource. Each cluster that fulfils these objectives will be defined as a *valid cluster* as defined below.

First we let $u'\in U(d_i,d_j)$ be a *clusterhead candidate* of the subset $U(d_i,d_j)$ iff $dist(u,u')\leq R_{max}$ for all $u\in U(d_i,d_j)$, where $dist(u,u')$ denotes the Euclidian distance between u and u', and R_{max} denotes the maximal transmission range.

Definition 1 The Boolean objective function F receives two inter-distances d_i,d_j, which form the subset $U(d_i,d_j)$, and

[1] This application warns a vehicle if it is attempting to merge from a location with limited visibility and another vehicle is approaching and predicted to occupy the intended merging space.

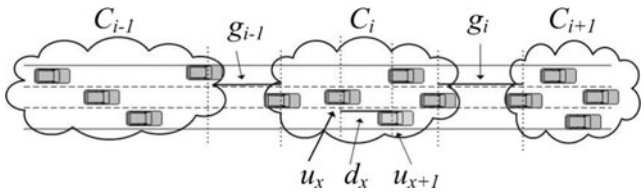

Fig. 2 The model basic notations

returns true if and only if this subset satisfies the following two conditions:

- $p \geq \min(p_{min}, k)$ where p is the number of clusterhead candidates in $U(d_i,d_j)$ and $k=|U(d_i,d_j)|$.
- $k \leq k_{max}$.

We note here that the D-CUT algorithm properties are preserved for any objective function that satisfies:

If $(U(d_i,d_j) \subseteq U(d_x,d_y)$ & $F(d_x,d_y)=true) \rightarrow F(d_i,d_j)=true$

(e.g., an objective function that allows some q hop connections between a clusterhead and its cluster members).

Based on this definition, we define a *valid solution* for the network N as follows.

Definition 2 Given the network N with the set of nodes $\{u_1,u_2,...,u_n\}$, the *Clustering Assignment* (CA) is a function assigning each node in the network to a cluster for which the received cluster set S fulfils: (i) every cluster in S satisfies the objective function; (ii) each node belongs to only one cluster; and (iii) the union of all clusters in S contains all nodes in the network.

At the performance topology level, we construct geographically optimized topology aims to provide an efficient and reliable inter-cluster bandwidth reuse. That is, grouping dense and consecutive nodes into clusters that are separated by maximally possible gaps. This type of clustering allows a strong connection between cluster members and reduces the inter-cluster interference. Having a *fairness* design goal in mind, we consider a *Max-Min* inter-cluster gap objective as the first objective of the optimization problem. In addition, in order to achieve efficient utilization of the allocated bandwidth, we consider minimizing the number of clusters in the network as the second objective of the optimization problem.

Let $V(N)$ be the set of all possible clustering assignments of the network. Now we are ready to formally define the optimal geographical clustering objectives described above:

- Objective 1: $\min_{i \in [1...m-1]} g_i$ is maximized over all solutions from $V(N)$.
- Objective 2: The number of clusters is minimized over all solutions from $V(N)$. Let us denote by S_{opt} the optimal solution such that $|S_{opt}| = \min_{S \in V(N)} |S|$.

The D-CUT algorithm produces the *Geographically Near-Optimal Clustering Assignment (GNOCA)* with the resulting cluster set S' that meets Objective 1 and approximates Objective 2 by a factor of 3.

4 The D-CUT algorithm

4.1 Overview

In this section we present the Distributed Construct Underlying Topology (D-CUT) algorithm. The D-CUT algorithm is an iterative algorithm, which strives to discover and maintain a geographically optimal clustering for the ever-changing network configuration. At each iteration the D-CUT algorithm gets a snapshot of the local vehicle proximity map and updates the clustering solution according to the changes in the network configuration.

The D-CUT algorithm exploits the constraint movement of vehicles (on roads) by basing the clustering scheme on road dissection. Specifically, clustering is achieved by partitioning the road into sections, where each section contains a different cluster. The D-CUT algorithm dissects the road by prioritizing the dissection candidate—the inter-distances—according to their size. By dissecting the road according to the inter-distance sizes, small scale (intra-cluster) reconfiguration changes are disregarded. Accordingly, as long as group of vehicles is traveling together, they will maintain their cluster form, even when intra-cluster changes have taken place. However, when larger scale changes occur, as groups of vehicles from different clusters noticeably approach each other, or alternately, subgroups of some cluster are considerably drifting apart, clustering reorganization will happen in order to maintain a geographically optimal clustering for the new network configuration. According to the above, the D-CUT algorithm consists of the following clustering reorganization procedures:

(i) The Split-Join procedure enables two groups of vehicles from adjacent clusters separated by a small inter-cluster gap and forming a valid cluster, to Join. So when two groups of vehicles approach each other up to the point at which the gap between them is smaller than the surrounding gaps, the Split-Join procedure is applied in order to create one valid cluster from these two nearby groups. To avoid unnecessary cluster reorganization, when one or two of the approaching groups is a sub-cluster, the sub-cluster is divided by a Split operation prior to the Join operation.

(ii) The Split procedure reacts to a scenario in which a cluster becomes invalid or discontinued; for example, in cases in which two groups within the same cluster drift apart. In this operation, the cluster will be divided at the maximal inner gap among the inner gaps forming two valid clusters.

(iii) The Join procedure is motivated by reducing the number of clusters in the model. Therefore, join conditions allow continuously increasing cluster size as long as this operation is not preventing more beneficial future operations. For this purpose, the Join conditions allow two clusters to join not only when a gap is located between two larger gaps as with the Split-Join procedure, but also when it is located between a larger gap from one side and non-joinable clusters from the other side.

4.2 Detailed description

Below is the formal explanation of this procedure:

Figure 3 presents the D-CUT algorithm run by vehicles that belong to cluster C_i. The algorithm uses as input the local vehicle proximity map of its vicinity. This map consists of: (i) the updated location of its own clusters and its two adjacent clusters (i.e., C_{i-1}, C_i, C_{i+1}) and (ii) the size of the inter-cluster gaps delimiting those clusters (i.e., g_{i-1}, g_i, g_{i+1}). In addition, the Join procedure requires two additional bits of information from each of its neighbor clusters, as will be detailed later. As output, the algorithm produces the new CA of C_i.

As mentioned above, the Split-Join procedure enables not only clusters but also sub-clusters to join. The following function is used to find the optimal cluster or sub-clusters, in terms of Objective 1, to be joined.

Definition 3 The *Max-Min Inter-Distance Pair* (*MMIDP*) is a function that finds a pair of inter-distances (denoted by $(d^{(l)}, d^{(r)})$) from adjacent clusters, such that the minimal value in the pair is maximized over all possible pairs forming a valid cluster. More formally, given the inter-cluster gap g_i, let $X = \{(d,d') | d \in D[g_{i-1}, g_i), d' \in D(g_i, g_{i+1}], F(d,d') = true\}$. The split candidates pair $(d^{(l)}, d^{(r)})$ is the pair that maximizes $\min(d,d')$ over all possible choices of $(d,d') \in X$. When more than one pair satisfies the condition, the pair with the maximal second pair value determines the unique *MMIDP*. In some cases we will refer to the output of the function $(d^{(l)}, d^{(r)})$ as *MMIDP*.

Definition 4 We define the following Split-Join Condition (*SJC*):

- $SJC(d^{(l)}, d^{(r)}, g_i) = \min(d^{(l)}, d^{(r)}) > g_i$.

Split-join procedure (Stages 1–2) Given the inter-cluster gap g_i, the Split-Join procedure (see Fig. 4a) enables two adjacent groups separated by this inter-cluster gap to join. To find the optimal cluster or sub-clusters to be joined, the procedure begins with finding the *MMIDP*, $(d^{(l)}, d^{(r)})$. Then, the *SJC* verifies whether $\min(d^{(l)}, d^{(r)})$ is larger than the inter-cluster gap, g_i, trapped between them. When this condition is satisfied, the Split-Join procedure removes the inter-cluster gap g_i by joining $U(d^{(l)}, g_i)$, $U(g_i, d^{(r)})$ to form the new cluster $U(d^{(l)}, d^{(r)})$. In case $U(d^{(l)}, g_i)$ is a sub-cluster (i.e., $d^{(l)} \neq g_{i-1}$), a preceding Split operation is applied on $d^{(l)}$, resulting in the additional cluster $U(g_{i-1}, d^{(l)})$. Symmetrically, when $d^{(r)} \neq g_{i+1}$, $U(d^{(r)}, g_{i+1})$ is formed. Only members of the new cluster $U(d^{(l)}, d^{(r)})$ terminate this iteration of the algorithm at the end of this stage, the rest continue to successive stages.

```
// Stage 1 - Split-Join procedure on g_{i-1}
(d^(l),d^(r)) = MMIDP(g_{i-1});
 if (SJC(d^(l),d^(r),g_{i-1}))
      if u∈U(d^(l),d^(r)) then C_{i-1}=U(d^(l),d^(r)) and exit;
      else C_i=U(d^(r),g_i);
// Stage 2 - Split-Join procedure on g_i
(d^(l),d^(r))= MMIDP(g_i);
 if(SJC(d^(l),d^(r),g_i))
      if u∈U(d^(l),d^(r)) then C_{i+1}=U(d^(l),d^(r)) and exit;
      else C_i=U(g_{i-1},d^(l));
// Stage 3 - apply Split procedure on C_i
if(SC1(C_i) || SC2(C_i))
     d'=max(D(g_{i-1},g_i)) where F(g_{i-1},d')= F(d',g_i)=true;
     if u∈U(g_{i-1},d') then C_i=U(g_{i-1},d') and exit;
     else C_{i+1}=U(d',g_i) and exit;
// Stage 4 apply Join procedure on g_{i-1}.
if(JC1(g_{i-1}) || JC2(g_{i-1})) && !(SJC(g_{i-2}) || SC2(C_{i-1})))
     C_i=U(g_{i-2},g_i);
// Stage 5 apply Join procedure on g_i.
if (JC1(g_i) || JC2(g_i)) && !(SJC(g_{i+1}) || SC2(C_{i+1})))
     C_i=U(g_{i-1},g_{i+1});
```

Fig. 3 The D-CUT algorithm

Each cluster first applies the procedure (stage 1) on its left inter-cluster gap and then (stage 2) on its right inter-cluster gap. Nevertheless, as we shall see in the following section, this procedure is performed in a coordinated fashion between the clusters. So when C_i applies a Split Join procedure with its left neighbor, C_{i-1} applies the same symmetric procedure with its right neighbor C_i. An example of the procedure is illustrated in Fig. 5.

Definition 5 We define the following Split Conditions:

- $SC1(C_i) = !F(g_{i-1}, g_i)$;
- $SC2(C_i) = d' > g_{i-1}, g_i$, where $d' = \max(D(g_{i-1}, g_i))$.

Split procedure (Stage 3) Given a cluster C_i and some inter-distance d', the split procedure is defined to partition the cluster C_i into two clusters: $U(g_{i-1}, d')$ and $U(d', g_i)$. In order to maintain stable CA, which consists of large clusters, the D-CUT tries to modify the current CA by a Split procedure only when the current CA contains clusters that are: (i) not satisfied by the objective function F, or (ii) discontinuous. Therefore,

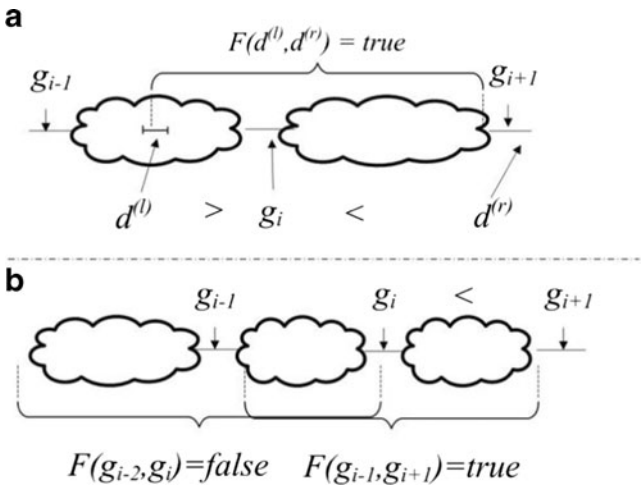

Fig. 4 **a** The Split Join procedure. In this example $SJC(d^{(l)},d^{(r)},g_i)$ is satisfied. As a result, a Split operation on $d^{(l)}$ is triggered, which is followed, at once, by a Join operation over g_i. Thus, the new *CA* of this range is the 2 clusters $U(g_{i-1},d^{(l)})$ and $U(d^{(l)},g_{i+1})$. **b** The Join procedure. Here, $JC2(g_i)$ is fulfilled. Consequently, a Join operation over g_i produces the new cluster $U(g_{i-1},g_{i+1})$

when a cluster ceases to satisfy the objective function *F*, the first split condition (*SC1*) is fulfilled. The second split condition (*SC2)* is satisfied when the inner gap becomes larger than its delimiting inter-cluster gaps. In both cases, the split operation is done on the maximal inter-distance that results in creation of two valid clusters.

Remark Due to the *SC1*, the D-CUT algorithm produces a valid *CA* at each iteration. An invalid *CA* will be received when the last iteration *CA,* updated by the new node's locations, creates one or more invalid clusters. When some of the clusters do not satisfy the objective function *F*, the Split operation, triggered by *SC1*, will occur. As a result, each invalid cluster is replaced by two[2] *valid* clusters. Since this operation is triggered independently among clusters, the split operations occur simultaneously, and a valid *CA* is received.

Definition 6 We define the following two Join Conditions:

- $JC1(g_i)=(g_{i-1}>g_i)\ \&\&\ !F(g_i,g_{i+2})\ \&\&\ F(g_{i-1},g_{i+1});$
- $JC2(g_i)=(g_{i+1}>g_i)\ \&\&\ !F(g_{i-2},g_i)\ \&\&\ F(g_{i-1},g_{i+1}).$

Join procedure (Stages 4–5) Given the gap g_i, the Join procedure (see Fig. 4b) is defined by removing the gap g_i to create the new cluster $U(g_{i-1},g_{i+1})$. The Join procedure is motivated by Objective 2, i.e., reducing the number of clusters in the model. Thus, the join conditions allow two clusters that form a valid cluster to join when it is trapped by a larger gap from one side and non-joinable clusters from the other side (*JC1* and *JC2*). To guarantee a coordinated Join operation, the Join conditions verify that the cluster to be joined with is not going to be split at the same iteration. Notice that this procedure requires two bits of additional information from each of its neighbor clusters. The first reflects whether its two immediate adjacent clusters are forming a valid cluster and the second is whether they satisfy the *SJC*.

4.3 Clusterhead election procedure

As mentioned, the high adaptivity of the algorithm to the dynamic environment leads to frequent and difficult to predict clusterhead changes. Though this quality is highly desired from a robustness point of view, it may lead to high overhead. Hereby, we present a low-cost, robust, and coordinated clusterhead election procedure. For this purpose, the D-CUT algorithm limits the clusterhead to a solitary function; aggregating the required information to form the updated vehicle proximity map and disseminating this map within the cluster. Each vehicle that successfully receives this map acquires all the required information to become the next iteration clusterhead.

Once a new cluster is formed by the algorithm, a clusterhead is elected based on the contention amongst the clusterhead candidates that are prioritized according to their proximity to the cluster center. The cluster center can be defined as center of gravity of graph representation of the cluster members. Only vehicles that successfully receive the coordinated proximity map will participate in the contention process. Thus, the contention sequence is agreed upon among the clusterhead candidates. By broadcasting a Clusterhead Declaration Message, a clusterhead candidate informs the rest of the cluster members that it is the current clusterhead.

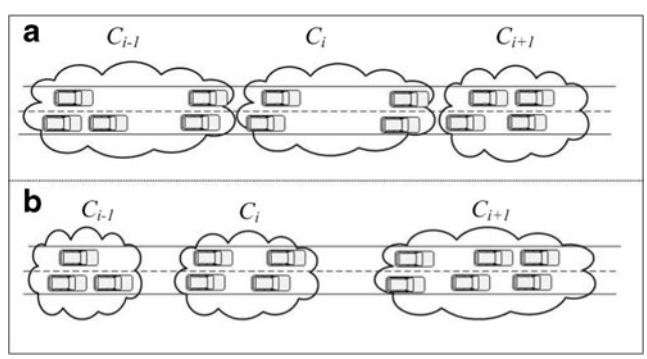

Fig. 5 An example of concurrent Split Join operation. **a** The input of the D-CUT algorithm for cluster C_i and (**b**) the output. In this scenario, since the last D-CUT execution, a group of two vehicles from cluster C_{i-1} have approached cluster C_i. At the same time, a group of two vehicles from cluster C_i gets very close to cluster C_{i+1}. The D-CUT will react to those changes by two parallel Split Join operations. By a Split Join procedure with its left neighbor, the right sub-cluster of C_{i-1} joins the left sub-cluster of C_i. By a Split Join procedure with its right neighbor, the right sub-cluster of C_i joins the cluster C_{i+1}

[2] Here we assume that an invalid cluster, which was a valid cluster in the previous iteration, can be split to 2 valid clusters. The algorithm can intuitively be expanded to deal with the case where an invalid cluster is required to be split to more than 2 clusters.

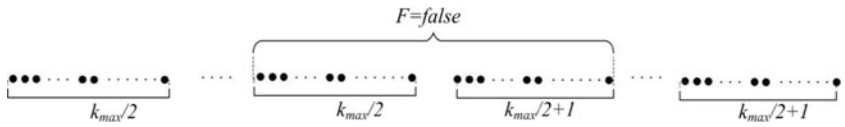

Fig. 6 Network configuration where the ratio between $|S_1|$ and $|S_2|$ converges to 2

The robustness of the contention increases with the increase in number of competitors. However, requiring a large number of competitors may lead to less-efficient clusters and longer contention period. The objective function-free parameter p_{min} enables balancing the robustness level and the clusters' efficiency. We present a short list of abbreviations in Appendix 1.

5 Theoretical analysis

Above we described a *CA* optimization problem based on the following two objectives: Objective 1 is to minimize the smallest inter-cluster gap over all solutions and Objective 2 is to minimize the number of clusters over all solutions. In some scenarios those two objectives contradict each other. To evaluate this effect, we set a lower bound approximation for our optimization problem. In particular, we show a lower bound for the approximation ratio for Objective 2, to every *CA* that satisfies Objective 1.

After showing this bound, we conduct a theoretical analysis to demonstrate the ability of the D-CUT algorithm to self-start and maintain an optimized *CA* in the dynamic VANET environment. For this purpose, we will show coordinate, local, and fast convergence of the algorithm to the *GNOCA*. This optimized *CA* meets Objective 1 and approximates Objective 2 by a factor of 3.

We first show a coordinate output when the algorithm is executed by the different vehicles. As mentioned above, the vehicle proximity map is distributed within the cluster from a single source and in one broadcast transmission. Thus, cluster-members that receive this map have the same input to the D-CUT algorithm, and thus, the same output. Adjacent clusters only share the overlapping portion of the vehicle proximity map, as each of them only shares information with the clusters directly adjacent to them. Despite the uncoordinated input, we will show that adjacent clusters have coordinated outputs. In particular, we will prove that if two adjacent clusters are involved in Split-Join or Join procedures, the procedures will be mirrored. This coordination is obtained from the overlapping portion of the coordinated map and with minimal coordination overhead.

We continue by demonstrating the locality of the D-CUT clustering process. In a properly functioning clustering algorithm, configuration changes in a certain place of the model will influence the clustering process of a *local* sub-model in the vicinity. Obviously, when the network is disconnected it automatically partitions into sub-networks. However, when considering a long and connected network as in a busy highway, the network in its entirety spans several kilometers. To demonstrate the locality of the D-CUT algorithm, we will show that the network is partitioned into sub-networks at certain inter-distances that are relatively larger than the inter-distances around them. Each such sub-model is clustered independently by the D-CUT algorithm.

To conclude, we show the fast and strict convergence of the D-CUT algorithm from any given *CA* to a *GNOCA*. An existing condition in showing convergence is the assumption that the configuration is stable (the exact definition of a stable configuration will be given later). This is clearly a weak assumption as the vehicular environment is very dynamic. Hence, in order to demonstrate the ability of the algorithm to deal with such a dynamic environment, we will show a very fast and strict convergence of the D-CUT algorithm. To prove the fastness and strictness, we will show the logarithmic convergence time solely of the distance between the initial *CA* and the *GNOCA* and not of the network in its entirety. This, combined with the fact that the D-CUT algorithm can be executed at a high rate, shows the ability of the D-CUT to converge even in the VANET's dynamic environment.

5.1 Lower bound

Hereby we show the lower bound for an approximation ratio for Objective 2, for every *CA* that satisfies Objective 1.

Theorem 1 There exists a network N so that any valid *CA* that meets Objective 1 approximates Objective 2 with factor of 2.

Proof We consider a network N organized in dense, equally spaced, groups of $k_{max}/2$ and $k_{max}/2+1$ nodes, where each group of $k_{max}/2$ nodes is followed by a group of $k_{max}/2+1$ nodes (see Fig. 6). Moreover, the inter-distances that separate the groups are larger than the inter-distances that separate the group members. Let us denote by S_1 and S_2 the *CA*s that meet Objective 1 and Objective 2, correspondingly. Under this configuration, the size of each cluster in S_2 is maximal, i.e., $|C|=k_{max}$ for $\forall C \in S_2$. Accordingly, $|S_2|=n/k_{max}$. On the other end, the *CA* that meets Objective 1, clusters each group into a cluster. Hence, $|S_1|=2 \cdot n/(k_{max}/2+k_{max}/2+1)$. Consequently, the ratio between $|S_1|$ and $|S_2|$ is 2.

5.2 Coordinate output

In this section we will show that the *CA* produced by the D-CUT algorithm is coordinated among all nodes. More formally, assume the output of D-CUT for some node u_x is C_p, and

for u_y is C_q; if $u_y \in C_p$ then $C_p = C_q$. Before proving the above assertion, let us establish the following:

Observation 1 In the case where $SJC(d^{(l)},d^{(r)},g_i)$ is satisfied: (i) if $d^{(l)} > d^{(r)}$ then $d^{(l)} = \max(D[d^{(l)},d^{(r)}])$ and $d^{(r)} = \max(D[g_i,d^{(r)}])$; (ii) symmetrically, if $d^{(l)} < d^{(r)}$, $d^{(r)} = \max(D[d^{(l)},d^{(r)}])$, and $d^{(l)} = \max(D[d^{(l)},g_i])$.

Proof Without loss of generality, let $d^{(l)} > d^{(r)}$. Since $SJC(d^{(l)},d^{(r)},g_i)$ holds, $d^{(l)},d^{(r)} > g_i$. Furthermore, from $d^{(r)}$ as the output of the *MMIDP* function, we can conclude that $d^{(r)} > d' \in D(g_i,d^{(r)})$, as $F(d^{(l)},d') = true$ (since $D(d^{(l)},d') \subset D(d^{(l)},d^{(r)})$). Symmetrically, $d^{(l)} > d \in D(d^{(l)},g_i)$.

From this observation we can conclude the following.

Observation 2 If $SJC(d_1^{(l)},d_1^{(r)},g_{i-1}) = true$ and $SJC(d_2^{(l)},d_2^{(r)},g_i) = true$ then $U(d_2^{(l)},d_1^{(r)}) = \phi$.

Lemma 1 Given that one of the Join conditions is satisfied on g_{i-1}, then g_i does not satisfy any of the Join conditions at the same iteration.

Proof Since g_{i-1} is satisfying one of the Join conditions we can conclude that $F(g_{i-2},g_i) = true$ and either $g_{i-1} < g_i$ or $F(g_{i-1},g_{i+1}) = false$. On the other hand, for g_i to satisfy the Join condition, the expression $F(g_{i-1},g_{i+1}) = true$ must be fulfilled and either $g_{i-1} > g_i$ or $F(g_{i-2},g_i) = false$. Thus, the lemma holds.

Theorem 2 Let the output of D-CUT for some node u_x be C_p and for u_y be C_q. If $u_y \in C_p$ then $C_p = C_q$.

Proof Consider some two nodes u_x and u_y, and let $C_{p(t)}, C_{q(t)}$ denote the clusters containing u_x and u_y, respectively, at any iteration t. In case $C_{q(t)}$ is not one of the adjacent clusters of $C_{p(t)}$, the new cluster $C_{p(t+1)}$ will not contain u_y since Split-Join and Join operations are applied only between adjacent clusters. In case $C_{p(t)} = C_{q(t)}$, the D-CUT algorithm will have the same input at iteration t for u_x and u_y, and therefore, it will produce the same output. Finally, consider the case that $C_{q(t)}$ is one of $C_{p(t)}$'s neighbors. Without loss of generality, let us assume that $C_{q(t)} = C_{p(t)-1}$. If $u_y \in C_{p(t+1)}$, the algorithm run by u_x performs either a Split-Join operation at stage 1 or a Join operation at stage 4. Following the symmetric nature between stage 1 and stage 2, if u_x performs a Split-Join operation in stage 1, then u_y performs an equivalent Split-Join operation at stage 2, as long as it is not a part of the newly formed cluster at stage 1. Observation 2 assures that u_y will not be a part of the newly formed cluster when $u_y \in C_{p(t+1)}$. Likewise, following the symmetry between stage 4 and stage 5, if u_x performs a Join operation in stage 4, then u_y performs an equivalent Split-Join operation in stage 5, as long as it does not perform any operation in the preceding four stages. If u_x performs a Join operation in step 4, according to the join conditions, no operation is performed by u_y in the first 3 stages. Finally, Lemma 1 assures that u_y is not involved in a Join operation at stage 4.

5.3 Independent sub-model clustering

In this section we seek to demonstrate the locality of the D-CUT clustering process. The algorithm, as we prove below, partitions the model N into local sub-models, where each sub-model is clustered *independently*. That is, configuration changes in one sub-model do not have any influence on *CA* changes in the rest of the model. This partition is done according to the local maximum inter-distance defined below:

Definition 7 Let $Q(d',t)$ be the set of any inter-distances d that satisfies either $F(d',d) = true$, or $F(d,d') = true$ at iteration t. We define d' as a *local maximum* inter-distance in the time frame $[t', t'']$, if and only if, $d' > \forall d \in Q(d',t)$ at any iteration t, $t' \le t \le t''$.

Now we shall confirm that as long as the two inter-distances remain the *local maximum* in the time interval $[t', t'']$, the sub-model trapped between them is clustered independently.

Definition 8 Let $g_{v(t)}$ be the inter-cluster gap located at the inter-distance d_v in iteration t, i.e., $d_v = g_{v(t)}$. Accordingly, $g_{v(t)-1}, g_{v(t)+1}$ are the 2 inter-cluster gaps that frame d_v from left and right, respectively, at iteration t.

Theorem 3 Consider d_v, d_u 2 consecutive local maximum inter-distances in the time frame $[t',t'']$. Then, the D-CUT algorithm is clustering the sub-network $U(d_v,d_u)$ independently of the rest of the model, in the time frame $[t'+1, t'']$.

Proof In order to establish this assertion, it is sufficient to show that the local maximum d_v partitions the network into 2 independently clustered, sub-networks. Notice that if d_v is not an inter-cluster gap at iteration t', the Split operation will take place on the local maximum d_v at iteration $t'+1$ Thus, d_v is an inter-cluster gap at iteration $t'+1$, i.e., $d_v = g_{v(t'+1)}$. From this iteration up to t'', Join operations on $d_v = g_{v(t)}$ are not viable, as Join conditions (*JC1,JC2*) require at least one of the inter-cluster gaps $g_{v(t)-1}, g_{v(t)+1}$ to be larger than the local maximum $g_{v(t)}$. By the same reasoning, a Join operation triggered by *SJC* will not take place, as this condition requires $d^{(l)},d^{(r)}$ to be larger than $g_{v(t)}$. Consequently, no operation between the adjacent clusters, which are separated by $g_{v(t)}$, will take place. To complete, we notice that $JC1(g_{v(t)-1}) = false$ regardless of whether or not $F(g_{v(t)-1}, g_{v(t)+1})$ is satisfied. Therefore, a Join operation on $g_{v(t)-1}$ is independent in *CA* in the range $D[g_{v(t)}, d_n]$. For reasons of symmetry, a Join operation on $g_{v(t)+1}$ is independent in the *CA* of the range $D[d_0, g_{v(t)}]$.

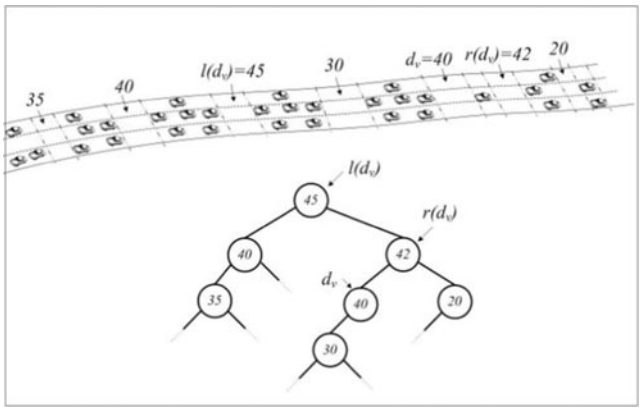

Fig. 7 The *SBT* of the set *D'= {35,40,45,30,40,42,20}*

5.4 Convergence process

In this section we show the fast and strict convergence of the D-CUT algorithm from any given *valid CA* to a *GNOCA*. In order to demonstrate the above, we will take advantage of the correlation between the D-CUT convergence processes and the Split Binary Tree (*SBT*), a particular tree representation of the inter-distance set D. Below, we analyze the convergence process by the following three stages: first, we present the *SBT* and prove that it is a Binary Search Tree with expected height of $O(\log(|D|))$; second, we limit the convergence time of the D-CUT algorithm by the height of the *SBT*; third, we express the *SBT* height as a function of the distance between the initial *CA* and the *GNOCA*.

5.4.1 The split binary tree (SBT)

In what follows, we refine the notation G to refer only to the local sub-model $D[d_s,d_f]$, i.e., $G(t)=G(t)\cap D[d_s,d_f]$. In addition, we refine the notation D to represent only the subset of the inter-distances that are involved in the convergence process. More formally, let D be the subset that contains all the inter-distances that at some iteration during the convergence process, served as a inter-cluster gap in the range $D[d_s,d_f]$. That is, $D=G(t_0)\cup G(t_0+1)\cup G(t_0+2)\ \ldots \cup G(t_0+t_2)$ where t_0,t_2 denote the first and last iterations in the conversance process, respectively.

Definition 9 Given a network N with configuration D, the Split Binary Tree *(SBT)* is a tree representation of the given configuration (see Fig. 7). The root entry of the *SBT* is associated with the full set $D(d_s,d_f)$. Each subsequent *SBT* entry is associated with the subset of D obtained by the following process: We start by setting d_k, the maximum inter-distance of the set $D(d_s,d_f)$, as the root entry. Then, we partition the set $D(d_s,d_f)$ into 2 subsets: $D(d_s,d_k)$, and $D(d_k,d_f)$, where the first subset is associated with the root's left child, and the second with the right child. Then, we set the maximum inter-distances d_y and d_z – where $d_y=\max(D(d_s,d_k))$, and $d_z=\max(D(d_k,d_f))$ – as the left and right children of d_k, respectively. We continue with this recursive process up to the point where each received subset contains a single inter-distance that obviously acts as its own maximum. As a key entry, we use the index of the maximum inter-distance (e.g., if d_v is the maximum distance in the entry, we set the key entry to v). By $l(d)$ and $r(d)$ we denote the left and right end points, respectively, of the associated range of d. Finally, the function $h(d_v)$ returns the height of the subtree rooted at the entry v.

Corollary 1 Given inter-distance set D, where D values are randomly distributed (i.e., form a random permutation), $SBT(D)$ produces a Random Binary Search Tree on the indices of the inter-distances with expected height of $O(\log|D|)$.

Proof Consider the *SBT(D)* produced by inserting the tree's entries in decreasing order. That is, we set the maximal inter-distance as the root. Then, at each stage we insert into the *SBT* the subsequent maximal value, which has not yet been inserted. We end when all inter-distances in D have been inserted. This *SBT* of the values of D is a Binary Search Tree considering the entry's keys (i.e., D indices). When the inter-distances' values are randomly distributed, this process inserts into the binary tree a random permutation of the keys set. Therefore, this process produces a Random Binary Search Tree on the indices of inter-distances. As demonstrated in

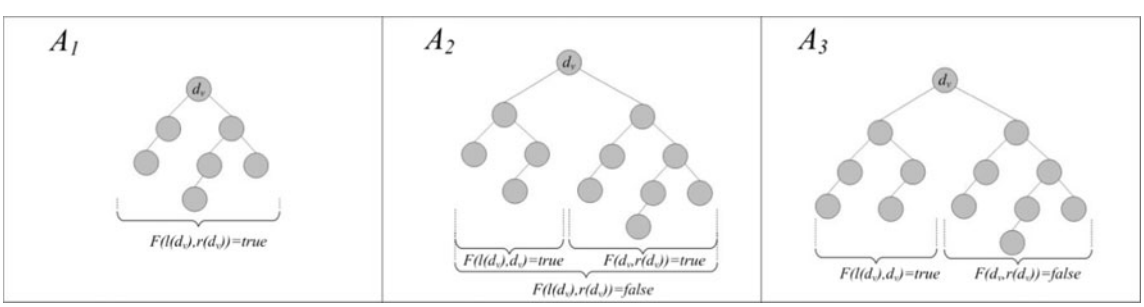

Fig. 8 Illustration of 3 of the 4 inter-distance types. The fourth type A_4 is a mirror image of A_3

[16], the expected height of such a *Random Binary Search Tree* is $O(\log |D|)$.

In order to show convergence, we need to assume *stable configuration*. Using the *SBT* representation of the given configuration, a more relaxed definition for stability can be achieved. We define a stable configuration as a configuration where: (i) d_s, d_f remain local maximums during all the convergence process and (ii) the *SBT* representation of the sub-model $D(d_s,d_f)$ is unchanged.

5.4.2 Bounding the convergence time by the SBT height

For bounding the convergence time by the height of *SBT*, we will show that each inter-distance is classified to its final state in the *GNOCA* according to its height in the *SBT*. But first, some additional definitions are required. We say that the inter-distance d is *classified* at iteration t' as an inner gap if $d \notin G(t)$ for every $t > t'$. We say that the inter-distance d is *classified* at iteration t' as an inter-cluster gap if $d \in G(t)$ for every $t > t'$. In addition, let us define a *refined height* $h'(d_v)$ of the sub-tree rooted at the entry v by counting only entries that will be classified as *inner gap*.

Following this, we associate each inter-distance with one of four inter-distance types (see Fig. 8).

Definition 10 Given $d_v \in D(d_s,d_f)$, we associate d_v according to the validity of the clusters trapped between $l(d_v)$, d_v, $r(d_v)$ as follows: (a) $d_v \in A_1$ if and only if $F(l(d_v),r(d_v)) = true$; (b) $d_v \in A_2$ if and only if $F(l(d_v),d_v)) = F(d_v,r(d_v)) = true$, and $F(l(d_v),r(d_v)) = false$; (c) $d_v \in A_3$ if and only if $F(l(d_v),d_v) = true$ and $F(d_v,r(d_v)) = false$; (d) $d_v \in A_4$ if and only if $F(l(d_v),d_v) = false$ and $F(d_v,r(d_v)) = true$.

Remark The final case where $F(l(d_v),d_v)) = F(d_v,r(d_v)) = false$ is already defined as the local maximum, i.e., the two sub-model end points.

The next observation shows the relationship between inter-distance type and the type of its *descendants* in the *SBT*.

Observation 3 Consider some inter-distance d_v; if $F(l(d_v),d_v) = true$ then all $d \in D(l(d_v),d_v))$ belong to A_1.

Proof Since every $d_i \in D(l(d_v),d_v))$ is smaller than both d_v, $l(d_v)$, we can deduce that $D(l(d_i),r(d_i)) \subseteq D(l(d_v),d_v))$. Thus, $F(l(d_i),r(d_i)) = true$.

From this observation we can conclude that if $d \in \{A_1 \cup A_2\}$ then all d descendants belong to A_1. Furthermore, the left *descendants* of $d \in A_3$ and the right descendants of $d \in A_4$ belong to A_1 as well.

Next, we will show the bottom-up classification process on the *SBT*. This process begins with inter-distances associated with A_1 that are placed (if they exist) in the bottom of the *SBT*. Lemma 2 assures that every $d \in A_1$ is classified as an inner gap at iteration $t = h'(d)$. Lemma 3 shows that the *CA* obtained at the end of this phase satisfies Objective 1. Then, in Lemma 4 we ensure that $d \in A_2$ is classified as an inter-cluster gap once its descendants, which are all associated with A_1, are classified. Notice that this condition is fulfilled at iteration $t = h'(d)$. We continue with the bottom-up process by demonstrating (Lemma 5) that $d \in \{A_3 \cup A_4\}$ is classified either as an inner gap or as an inter-cluster gap at iteration $t = h'(d)$. To conclude, we prove that after the classification of all $d \in D[d_s,d_f]$ the obtained *CA* is in fact the *GNOCA* (Lemma 6). Note that the sub-model end-points d_s, d_f are classified as inter-cluster gaps at iteration t_0 as we have shown in the proof of Theorem 3. Appendix 2 provides a simple example of the relation between the *SBT* height and the convergence time.

In order to demonstrate the classification of inter-distance d as an inner gap, we will ensure that if $d \in G(t)$ at iteration $t = h'(d)$, then the Join operation will be applied on d. In case $d \in A_1$ we will demonstrate that *SJC* is satisfied, and when $d \in \{A_3, A_4\}$ the operation will be triggered by *JC1* or *JC2*. However, to guarantee the classification, we need to prove that this operation will not be overturned by a future Split operation. To this end, in the following observations we will set some conditions that the Split operation can fulfill. First, we will show that *SC2* is satisfied only on the maximal inter-distance within a cluster (Observation 4) and only at iteration t_0 (Observation 5). Second, we will show that if some *SBT* sub-tree does not contain any inter-distance that plays the role of inter-cluster gap at iteration t, then the Split operation will not be applied on any inter-distance in this sub-tree, at any iteration $t' > t$ (Observation 6).

Remark As we assume a valid *CA* at iteration t_0, and as all the D-CUT operations produce valid clusters, in the following we assume the Split operation to be triggered either by *SC2* or by *SJC*.

Observation 4 If $SC2(C_i)$ is satisfied on d' then $d' = \max(D[g_{i-1},g_i])$.

Proof Follows directly from the definition of *SC2*.

Observation 5 Let $g_{i(t)-1}, g_{i(t)}$ be two consecutive inter-cluster gaps at iteration t. For every $t > t_0$, all $d \in D(g_{i(t)-1}, g_{i(t)})$ are smaller than $\max(g_{i(t)-1}, g_{i(t)})$.

Proof (Proof by contradiction). Let us assume the opposite; that is, there exists $d_x \in D(g_{i(t)-1}, g_{i(t)})$ that satisfies $d_x > \max(g_{i(t)-1}, g_{i(t)})$. Since $g_{i(t)-1}, g_{i(t)}$ are two consecutive inter-cluster gaps at iteration t we can deduce that d_x does not satisfy *SC2* at iteration $t-1$. Therefore, either $d_x \in G(t-1)$ and a Join operation takes place on d_x at iteration $t-1$, or $d_x \notin G(t-1)$ and $\max(g_{i(t-1)-1}, g_{i(t-1)}) > d_x$. However, regarding the first option, the Join operation on d_x results in $\max(g_{i(t)-1}, g_{i(t)}) > d_x$,

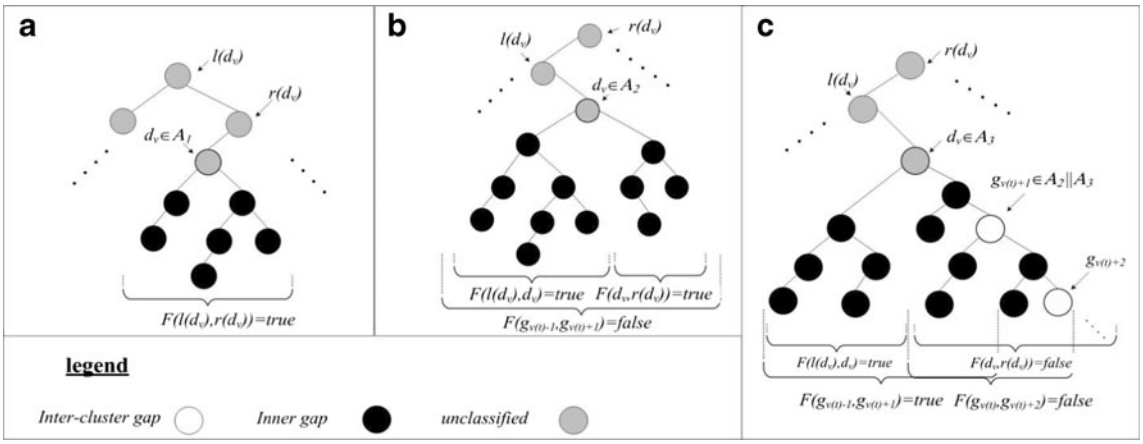

Fig. 9 *SBT* state at the iteration of d_v classification: **a** $d_v \in A_1$,: all d_v descendants have been classified at $t=h'(d_v)$ as inner-gaps. Hence, $SJC(l(d_v),r(d_v),g_v)$ holds, and d_v is classified as inner-gap; **b** $d_v \in A_2$: in this case as well, d_v descendants have been classified at $t=h'(d_v)$. However, as $F(l(d_v),r(d_v))=false$, d_v is classified as an inter-cluster-gap; **c** $d_v \in A_3$: this type of inter-distance can be classified either as an inner-gap (as illustrated here) or as an inter-cluster-gap. This is determined according to the outcome of the $F(g_{v(t)-1},g_{v(t)+1})$ at $t=h'(d_v)$

which contradicts our initial assumption. Thus, we are left with the second option in which at iteration $t-1$, $\max(g_{i(t-1)-1},g_{i(t-1)})>d_x$. For reasons of symmetry, let us assume that $g_{i(t-1)-1}<g_{i(t-1)}$. Accordingly, to satisfy $d_x>\max(g_{i(t)-1},g_{i(t)})$, $g_{i(t-1)}$ is replaced by $g_{i(t)}$, where $g_{i(t)}<g_{i(t-1)}$. The replacement of $g_{i(t-1)}$ by the smaller $g_{i(t)}$ cannot be the outcome of a Join operation on $g_{i(t-1)}$, since only a Join operation triggered by $JC1(g_{i(t-1)})$ can result in $g_{i(t-1)}>g_{i(t-1)+1}$, $g_{i(t-1)+1}=g_{i(t)}$, and $JC1(g_{i(t-1)})=false$. Therefore, we are left with the case where $g_{i(t)}$ is created by a Split operation in the range between d_x and $g_{i(t-1)}$. As *SC2* is applied only on an inter-distance larger than its framing inter-cluster gaps, this Split operation is triggered by $SJC(d^{(l)},d^{(r)},g_{i(t-1)-1})$, where $g_{i(t)}=d^{(r)}$. In such case, $d_x \in D(d^{(l)},d^{(r)})$, and therefore, $F(d^{(l)},d_x)=true$. Thus, $g_{i(t)}=d^{(r)}>d_x$ as the pair $(d^{(l)},d^{(r)})$ is preferred over $(d^{(l)},d_x)$ by the *MMIDP* function. This contradicts our initial assumption that $d_x>\max(g_{i(t)-1},g_{i(t)})$.

Observation 6 If $D(l(d_v),r(d_v)) \cap G(t') = \phi$, then $D(l(d_v),r(d_v)) \cap G(t) = \phi$ for every $t>t'$.

Proof To establish this observation it suffices to show that if $D(l(d_v),r(d_v)) \cap G(t') = \phi$, then $D(l(d_v),r(d_v)) \cap G(t'+1) = \phi$. Explicitly, a Split operation is not taking place at iteration t' in the range $D(l(d_v),r(d_v))$. From $D(l(d_v),r(d_v)) \cap G(t') = \phi$, it can be deduced that $D[l(d_v),r(d_v)] \subseteq D[g_{i(t)-1},g_{i(t)}]$, where $g_{i(t)-1},g_{i(t)}$ are the two inter-cluster gaps that frame d_v from left and right, respectively. Accordingly, neither *SJC* (Observation 1) nor *SC2* (Observation 4) hold, since by definition, both $l(d_v),r(d_v)$ are larger than every $d \in D(l(d_v),r(d_v))$.

After setting the conditions under which the Split operation can be fulfilled, we begin describing the bottom-up classification process by demonstrating the classification of $d_v \in A_1$ as the inner gap at iteration $t=h'(d_v)$.

Lemma 2 If $d_v \in A_1$, then d_v is classified at iteration $t=h'(d_v)$, as an inner gap.

Proof We will demonstrate this lemma by way of induction on the inter-distance refined height. In the base case, d_v ($d_v=g_{v(t0)}$) is an *SBT* leaf ($h(d_v)=0$). In this case $D(l(d_v),d_v)=D(d_v,r(d_v))=\phi$, and therefore, $l(d_v) \in D[g_{v(t0)-1}, g_{v(t0)})$ and $r(d_v) \in D(g_{v(t0)-1}, g_{v(t0)+1}]$. In addition, since $d_v \in A_1$, we can conclude that $F(l(d_v),r(d_v))=true$, where by definition $l(d_v),r(d_v)>g_{v(t0)}$. Consequently, $SJC(l(d_v),r(d_v),g_{v(t0)})$ is satisfied and a Join operation on $g_{v(t0)}$ will take place at iteration t_0. Subsequent to this operation, $D(l(d_v),r(d_v)) \cap G(t_0+1) = \phi$. According to Observation 6, this assures that d_v is classified at iteration t_0 as the inner gap as a Split operation on d_v will not take place at any $t>t_0$. For the inductive step, let us assume that the lemma holds for all d such that $h'(d) \leq t-1$. Accordingly, as all d_v descendants belong to A_1 (Observation 3), they all have been classified as inner gaps at iteration $t-1$. Therefore, $D(l(d_v),d_v) \cap G(t-1)=D(d_v,r(d_v)) \cap G(t-1)=\phi$ (see Fig. 9a). As demonstrated in the base case, this assures the classification of d_v as an inner gap.

Lemma 3 Let $t_1=t_0+\max(h(d))+1$, for all $d \in D(d_s,d_f) \cap A_1$. $G(t)$ satisfies Objective 1 for every $t \geq t_1$.

Proof Let us denote by $d_v=g_{v(t)}$ the minimal inter-cluster gap in $G(t)$ at some iteration $t \geq t_1$. According to Lemma 2, $d_v \notin A_1$, and thus, $F(l(d_v),r(d_v))=false$. This implies that every valid *CA* must contain at least one inter-cluster gap in the

range $D(l(d_v),r(d_v))$. The lemma holds as d_v is the maximal inter-distance in this range.

The meaning of this result is that if we subdivide the model at any inter-distance $d \notin A_1$, each such sub-model satisfies Objective 1.

Now we wish to continue with the bottom-up classification process by demonstrating the classification of $d_v \in A_2$ as an inter-cluster gap at iteration $t = h'(d_v)$. In order to demonstrate the classification of d as an inter-cluster gap, we will show that d is located between two clusters, and their union produces an invalid cluster. Considering such d, and assuming that all d descendents from A_1 are classified as inner gap, the following observation ensures that this state is irreversible.

Observation 7 Consider $d_v \notin A_1$ ($d_v = g_{v(t)}$). If $F(g_{v(t)-1}, g_{v(t)+1}) = false$ at some iteration $t \geq h'(d_v)$, then $F(g_{v(t)-1}, g_{v(t)+1}) = false$ at any iteration $t' \geq t$.

Proof The observation follows if a Split operation will not take place in the range $D(g_{v(t)-1}, g_{v(t)+1})$ at any $t' \geq t$. From Observation 5 we can conclude that *SC2* is applied only at iteration t_0. In addition, since (i) only $g \in A_1$ can satisfy $SJC(d^{(l)}, d^{(r)}, g)$, and (ii) all d_v descendants that belong to A_1 are classified as inner gap at iteration t (Lemma 2), a Split operation in the range $D(g_{v(t)-1}, g_{v(t)+1})$ will not be triggered by any $g \in D(l(d_v), r(d_v))$. To conclude, we note that any $g \notin D(l(d_v), r(d_v))$ will not trigger a Split operation (by satisfying *SJC*) on $d \in D(l(d_v), r(d_v))$. This is because either $l(d_v)$ or $r(d_v)$ are located in the range between d and g, such that $\min(l(d_v), r(d_v)) > d$. As demonstrated in Observation 1, in such a case, the Split operation will not be applied on d.

Lemma 4 If $d_v \in A_2$, then d_v is classified as an inter-cluster gap at iteration $t = h'(d_v)$.

Proof Following Observation 3, in case $d_v \in A_2$, all its descendants (if any exist) are from A_1. Following Lemma 2, this implies that all d_v descendants are classified as inner gaps at iteration t (see Fig. 9b). Hence, from $F(l(d_v), r(d_v)) = false$ we can conclude that (i) $d_v = g_{v(t)} \in G(t)$ and (ii) $F(g_{v(t)-1}, g_{v(t)+1}) = false$. The lemma follows from Observation 7, which ensures that $F(g_{v(t')-1}, g_{v(t')+1}) = false$ at any iteration $t' > t$.

To complete the bottom-up classification process, we wish to show the classification of $d_v \in \{A_3 \cup A_4\}$ at iteration $t = h'(d_v)$. Lemma 5 confirms that if d_v is located between non-joinable clusters (i.e., their union produces an invalid cluster) at iteration t, then it is classified as an inner gap; while if d_v is located between joinable clusters, either *JC1* (if $d_v \in A_3$) or *JC2* (if $d_v \in A_4$) will be satisfied on d_v, and as a result, it will be classified as an inner gap. To show that indeed the Join conditions are satisfied, in the following two observations we characterize the inter-cluster gaps $g_{v(t)-1}, g_{v(t)+1}$, framing $d_v = g_{v(t)}$ from left and right, respectively, at iteration t.

Observation 8 Let $t' = h'(d_v)$. If $d_v \in A_3$, then $g_{v(t)-1} > g_{v(t)}$ for every $t \geq t'$.

Proof Since $d_v \in A_3$ then all d_v left descendants (i.e., $\forall d \in D(l(d_v), d_v)$) are from A_1 (Observation 3). Thus, all d_v left descendants are classified as inner gaps at iteration t' (Lemma 2). Therefore, $l(d_v) \in D[g_{v(t)-1}, g_{v(t)})$ for every $t > t'$. Obviously, if $l(d_v) = g_{v(t)-1}$, the lemma follows as $l(d_v) > g_{v(t)}$. On the other hand, if $l(d_v) \in D(g_{v(t)-1}, g_{v(t)})$, following Observation 5 $l(d_v) < \max(g_{v(t)-1}, g_{v(t)})$. Since $l(d_v) > g_{v(t)}$ we can conclude that $g_{v(t)-1} > g_{v(t)}$.

Observation 9 Let $t' = h'(d_v)$. If $d_v \in A_3$ then $g_{v(t)+1} \in \{A_2 \cup A_3\}$ for every $t \geq t'$.

Proof According to Lemma 2, $g_{v(t)+1} \notin A_1$ at any iteration $t \geq t'$. In addition, since $d_v \in A_3$, by definition, $F(d_v, r(d_v)) = false$. Therefore, $g_{v(t)+1}$ is a descendant of d_v, and thus, $d_v > g_{v(t)+1}$. Since $F(d_v, g_{v(t)+1}) = true$ we can deduce that $g_{v(t)+1} \notin \{d_f \cup A_4\}$. Thus, $g_{v(t)+1} \in \{A_2 \cup A_3\}$ at any iteration $t \geq t'$.

Lemma 5 If $d_v \in A_3$ then d_v is classified at iteration $t = h'(d_v)$ either as an inner gap when $F(g_{v(t)-1}, g_{v(t)+1}) = true$ or as an inter-cluster gap when $F(g_{v(t)-1}, g_{v(t)+1}) = false$.

Proof Following Observation 7, if $d_v \in A_3$ and $F(g_{v(t)-1}, g_{v(t)+1}) = false$, then d_v is classified as an inter-cluster gap. Hence, to prove the lemma, we will ensure that if $d_v \in A_3$ and $F(g_{v(t)-1}, g_{v(t)+1}) = true$, then the Join operation, triggered by $JC1(g_{v(t)})$, occurs at iteration t. We will demonstrate it by induction on the inter-distance refined height. By Observation 8 we get that $g_{v(t)-1} > g_{v(t)}$ at iteration t. Thus, to show that $JC1(g_{v(t)})$ is satisfied it is sufficient to show that $F(g_{v(t)}, g_{v(t)+2}) = false$ (see Fig. 9c). To simplify the notation, we let $d_v \in A_3{}^*$ if $d_v \in A_3$ and $F(g_{v(t)-1}, g_{v(t)+1}) = true$.

For the base case, we consider the d_v without descendant from $A_3{}^*$. Notice that $g_{v(t)+1}$ is a $g_{v(t)}$ descendant and belongs to $\{A_2, A_3\}$ (Observation 9). According to Lemma 4, if $g_{v(t)+1} \in A_2$ then $F(g_{v(t)}, g_{v(t)+2}) = false$. If $g_{v(t)+1} \in A_3$, $F(g_{v(t)}, g_{v(t)+2}) = false$ as d_v has no descendant from $A_3{}^*$. Therefore, $JC1(g_{v(t)})$ is satisfied. Following Observation 6, a Split operation on d_v will not be applied at any iteration $t' > t$, and thus, d_v is classified as an inner gap. Assume that our induction hypothesis holds for all d such that $h'(d) \leq t-1$. Here as well, the case where $g_{v(t)+1} \in A_2$ comes from Lemma 4. If $g_{v(t)+1} \in A_3$, $F(g_{v(t)}, g_{v(t)+2}) = false$ directly follows the inductive hypothesis. As in the base case, the assertion is concluded by Observation 6.

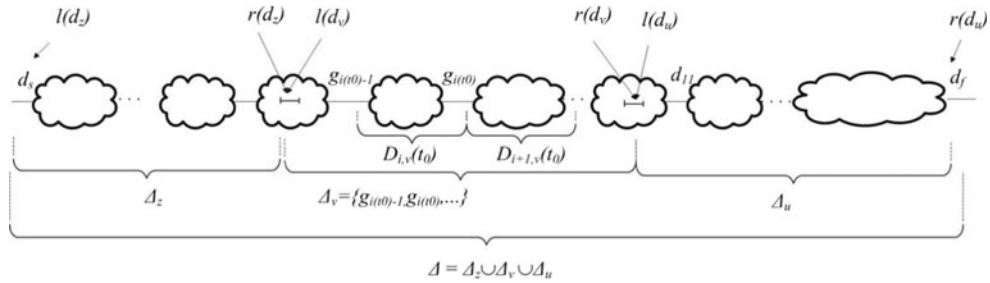

Fig. 10 Notations used in the D-CUT strict convergence subsection

For reasons of symmetry the above lemma holds for $d_v \in A_4$.

Lemma 6 Let $t_2 = t_0 + \max(h'(d)) + 1$ for all $d \in D(d_s, d_f)$. $G(t)$ satisfies Objective 2 with an approximation ratio of at most 3 for every $t \geq t_2$.

Proof In order to compare the values of Objective 2 in optimal *CA* and the *CA* produced by the D-CUT algorithm, we will bound the number of inter-cluster gaps in each sub-model range separately. As the sub-model $D[d_s, d_f]$ shares the endpoint d_s with its left sub-model and d_f with its right sub-model, we count only the left endpoints in each sub-model. To be exact, we determine $|G(t_2)|-1$ inter-cluster gaps for the sub-model $D[d_s, d_f]$. As demonstrated above (in the proofs of Lemma 4 and Lemma 5), each inter-cluster gap $g_{i(t2)} \in G(t_2)$, excluding the sub-model end points $\{g_{s(t2)}, g_{f(t2)}\}$, satisfies $F(g_{i(t2)-1}, g_{i(t2)+1}) = false$. Accordingly, $G(t_2)$ can be segmented into $\lfloor {}^{1}/_{2}|G(t_2)|-1 \rfloor$ pairs of consecutive clusters, where the union of each cluster pair produces an invalid cluster. (In case of an odd number of clusters we remain with the right-most cluster unpaired.) This implies that every valid *CA* has at least $\lfloor {}^{1}/_{2}|G(t_2)|-1 \rfloor$ inter-cluster gaps in the range $D[d_s, d_f]$, because every valid *CA* contains at least one inter-cluster gap in the range of each consecutive pair of clusters. Let G_{opt} be the set of the inter-cluster gaps in optimal *CA* (in the perspective of term 2) in the range $D[d_s, d_f]$. We obtain that $|G(t_2)|-1 \leq 2\,|G_{opt}|+1$. Since $F(d_s, d_f) = false$, $|G_{opt}| \geq 1$. Hence, in the worst case we get an approximation ratio of 3. The ratio converges to the lower bound 2 in approximation with the increase of sub-model size.

Following the above lemmas (Lemmas 2 through 6), we can establish the following intermediate conclusion:

Corollary 2 The D-CUT algorithm converges to the *GNOCA* after no more than t_2 iterations.

5.5 D-CUT strict convergence

After we have limited the convergence time by the *SBT* refined height, we want to express the *SBT* refined height as a function of the distance between the initial *CA* and the *GNOCA*, i.e., $|(G(t_0) \cup G(t_2)) \setminus (G(t_0) \cap G(t_2))|$. We consider only the following subset to express the refined height of the *SBT:*

Definition 11 Let $\Delta = G(t_0) \setminus G(t_2)$ be the set of inter-cluster gaps in the range $D(d_s, d_f)$ that belong to the initial *CA* $G(t_0)$, but do not belong to the *GNOCA*, $G(t_2)$. (Obviously, $\Delta \subseteq (G(t_0) \cup G(t_2)) \setminus (G(t_0) \cap G(t_2))$.)

Definition 12 Let $\Psi = D(d_s, d_f) \setminus (G(t_0) \cup G(t_2))$ be the set of temporary inter-cluster gaps in the range $D(d_s, d_f)$ that appears (by Split operation), and is then removed (by Join operation) during the course of the convergence process.

Notice that the union of the sets Δ and Ψ gives the set of all inter-distances that are classified as inner gaps. As the refined height is a function of the inter-distances that are classified as inner gaps, and Δ is a lower bound of the distance between the initial *CA* and the *GNOCA*, our goal is to express the ratio between the size of sets Δ and $\Psi \cup \Delta$.

Below, we demonstrate that $|\Psi \cup \Delta| \leq 3.5 \cdot |\Delta|$ by showing that $|\Psi| \leq 2.5 \cdot |\Delta|$. In order to set this bound, we will relate all Split operations that occur during the convergence process to an explicit subset of Δ. In particular, we define the subset Δ_v to be the set of inter-cluster gaps that are located in the range $D(l(d_v), r(d_v))$ at iteration t_0, i.e., $\Delta_v = D(l(d_v), r(d_v)) \cap G(t_0)$, where $d_v \in A_1$ and both $l(d_v), r(d_v) \notin A_1$. Since every $d \in D(l(d_v), r(d_v))$ belongs to A_1, we can conclude that every such d is classified as an inner gap (Lemma 2), and thus, $\Delta_v \subseteq \Delta$. The right neighbor subset of Δ_v is denoted by $\Delta_u = D(r(d_v), r(d_u)) \cap G(t_0)$ (see Fig. 10).

First, we relate each Split operation that takes place in the range $D[l(d_v), r(d_v)]$ to the subset Δ_v. This criterion is sufficient to relate any Split operation triggered by *SC2* to one of Δ subsets. To see why, recall that according to Observation 6, if $\Delta_v = \phi$, then the Split operation will not take place in the range $D(l(d_v), r(d_v))$. Moreover, in Observation 9 we show that if $\Delta_v = \phi$ and a Split operation, triggered by *SC2*, takes place on $r(d_v)$, then $\Delta_u \neq \phi$. Accordingly, such a Split operation can be related either to Δ_v (if $\Delta_v \neq \phi$) or to Δ_u (if $\Delta_u \neq \phi$). However, when considering a Split operation triggered by *SJC*, the above criterion is not enough. This is because Split operation triggered by $SJC(d^{(l)}, d^{(r)}, g_i)$ can be spread outside the range $D[l(d_v), r(d_v)]$ even if $g_i \in D[l(d_v), r(d_v)]$. In Observation 10 we demonstrate that in such a case the new inter-cluster gap

does not belong to A_1. Thus, this new inter-cluster gap will not trigger an additional Split operation triggered by *SJC*. Hence, by relating any Split operation triggered by $SJC(d^{(l)},d^{(r)},g_i)$, where $g_i \in D[l(d_v),r(d_v)]$, to the subset Δ_v we ensure that any Split operation triggered by *SJC* will be related to one of Δ subsets.

Definition 13 We say that a Split operation on *d results* in Δ_v. $\Delta_v \neq \phi$, if one of the following is satisfied: (i) $d \in D[l(d_v),r(d_v)]$ or (ii) $d \notin D[l(d_v),r(d_v)]$, and the operation is triggered by the inter-cluster gap g_i, where $g_i \in D[l(d_v),r(d_v)]$.

Observation 9 Let Δ_v, Δ_u be two adjacent Δ subsets. Let $d' = r(d_v) = l(d_u)$. If *SC2* is satisfied on d' at iteration t_0, then either $\Delta_v \neq \phi$ or $\Delta_u \neq \phi$.

Proof Without loss of generality let $d' \in A_2 \cup A_3$. We will show that $\Delta_v \neq \phi$. In this case $l(d') = l(d_v)$, since by definition the ranges $D(l(d'),d')$, $D(l(d_v),d')$ contain only $d \in A_1$, and both $l(d'), l(d_v) \notin A_1$. Hence, to demonstrate the observation we will show that $G(t_0) \cap D(l(d'),d') \neq \phi$. Let $g_{j(t0)-1}, g_{j(t0)}$ be the two inter-cluster gaps framing d' at iteration t_0 from the left and right, respectively, i.e., $d' \in D(g_{j(t0)-1}, g_{j(t0)})$. According to Observation 4, $d' = \max(D[g_{j(t0)-1}, g_{j(t0)}])$. In addition, by definition $l(d') > d'$. Thus, $l(d') \notin D[g_{j(t0)-1}, g_{j(t0)}]$, i.e., $g_{j(t0)-1} \in D(l(d'),d')$.

Observation 10 Consider the case when $SJC(d^{(l)},d^{(r)},g_i)$ is satisfied. If $g_i \in D(l(d_v),r(d_v))$ and $d^{(l)} \in A_1$, then $d^{(l)} \in D(l(d_v),r(d_v))$.

Proof (Proof by contradiction.) Assume to the contrary that $d^{(l)} \notin D(l(d_v),r(d_v))$. Accordingly, $l(d_v) \in D(d^{(l)},g_i)$. In addition, since $d^{(l)} \in A_1$ we can deduce that $r(d^{(l)}) \in D(d^{(l)},g_i)$ as well. As by definition $r(d^{(l)}) > d^{(l)}$ and $l(d_v) > g_i$, $\max(l(d_v),r(d^{(l)})) > \max(d^{(l)},g_i)$. According to Observation 1, in such a case $SJC(d^{(l)},d^{(r)},g_i) = false$.

According to the above, we can establish the inequality $|\Psi| \leq 2.5 \cdot |\Delta|$ by showing that the number of Split operations *resulting* from the set Δ_v is bounded by $2.5\ |\Delta_v|$.

We first consider the base case where $|\Delta_v| = 1$. In this case no more than 2 Split operations (on $l(d_v),r(d_v)$) will result from the Join operation on $g_{i(t0)}$. This is because $SJC(l(d_v),r(d_v),g_{i(t0)}) = true$ at iteration t_0. After the Join operation on $g_{i(t0)}$, the range $D(l(d_v),r(d_v))$ (which does not contain any inter-cluster gap) will not be split, as demonstrated in Observation 6.

Next, we show that if $|\Delta_v| \geq 2$, then no more than $2 \cdot |\Delta_v| + 1$ Split operations will result from the set Δ_v. As we seek an upper bound, we are allowed to assume that if $\Delta_v \neq \phi$ then $SJC(l(d_v),r(d_v),d_v)$ will be satisfied. Therefore, we presume that the Split on $l(d_v),r(d_v)$ will result in $\Delta_v \neq \phi$. According to the above, the total number of Split operations resulting from the set Δ_v is limited to the sum of: (i) the number of Split operations on the both ends of the sub-model: $l(d_v),r(d_v)$, (ii) the number of Split operations (either by fulfilling *SJC* or *SC2*) taking place in the range $D(l(d_v),r(d_v))$, and (iii) the number of Split operations taking place (by fulfilling $SJC(d^{(l)},d^{(r)},g_i)$) outside the range $D[l(d_v),r(d_v)]$, where $g_i \in D(l(d_v),r(d_v))$.

In the following two observations we will characterize the split candidates in the range $D(l(d_v),r(d_v))$ according to the initial *CA* at this range. To this, we let $D_{v,j}(t) = D[l(d_v),r(d_v)] \cap D[g_{j(t)-1},g_{j(t)}]$.

Observation 12 If $SJC(d^{(l)},d^{(r)},g_{i(t)})$ is satisfied, and both $d^{(l)}, g_{i(t)} \in D_{v,i}(t)$ at iteration t, then $d^{(l)} = \max(D_{v,i}(t))$.

Proof We subdivide this proof into two cases according to $d^{(r)}$ position. (i) In case $d^{(r)} \in D(g_{i(t)},r(d_v)]$, $F(d,d^{(r)}) = true$ for all $d \in D_{v,i}(t)$ as $F(l(d_v),r(d_v)) = true$. Thus, according to Observation 1, $d^{(l)} = \max(D_{v,i}(t))$. (ii) In case $d^{(r)} \in D(r(d_v),g_{i(t)+1}]$, i.e., $r(d_v) \in D(g_{i(t)},d^{(r)})$, $F(d^{(l)},r(d_v)) = true$. Hence, $d^{(r)} > r(d_v)$ as the pair $(d^{(l)},d^{(r)})$ is preferred over $(d^{(l)},r(d_v))$ by the *MMIDP* at iteration t. In addition, since by definition $r(d_v)$ is larger than all $d \in D_{v,i}(t)$ we can conclude that $d^{(l)} < \min(r(d_v),d^{(r)})$. The assertion follows as $(d^{(l)},d^{(r)})$ is preferred over $(d,r(d_v))$ for all $d \in D_{v,i}(t)$.

For reasons of symmetry, the above observation holds for $d^{(r)}$ as well. In Observation 12 we demonstrated that a split candidate d must satisfy $d = \max(D_{v,i}(t))$. In the following we extend this split candidate prerequisite to $d = \max(D_{v,i}(t_0))$.

Observation 13 If $d \neq \max(D_{v,j}(t_0))$ then $d \neq \max(D_{v,j}(t))$ for every $t \geq t_0$.

Proof In order to have $d = \max(D_{v,j}(t+1))$, when $d \neq \max(D_{v,j}(t))$, a Split operation in the range between d and $\max(D_{v,j}(t))$ is required. However, all inter-distances in this range are smaller than $\max(D_{v,j}(t))$. According to the above observations, such an operation will not take place at iteration t as neither *SC2* (Observation 4) nor *SJC* (Observation 12) is satisfied on d'.

After stating the above, we are ready to limit the number of Split operations resulting in the set Δ_v. In Lemma 8, we limit the number of Split operations on $d \in D(l(d_v),r(d_v))$, resulting from Δ_v, by $|\Delta_v| - 1$. Lemma 9 assures that the maximal number of Split operations resulting from Δ_v, on $d \notin D[l(d_v),r(d_v)]$ is $|\Delta_v|$.

Lemma 8 The maximal number of Split operations on $d \in D(l(d_v),r(d_v))$ resulting from the set Δ_v is $|\Delta_v| - 1$.

Proof As *SC2* is applied only at iteration t_0 (Observation 5) and only on the maximal inter-distance in the cluster

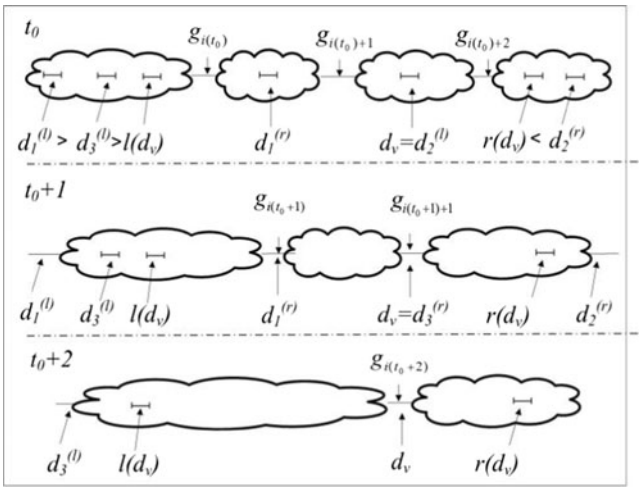

Fig. 11 Illustration of the worst case scenario (in terms of Split operations number) resulting in the set $\{g_{i(t0)}, g_{i(t0)+1}, g_{i(t0)+2}\}$. At t_0, $SJC(d_1^{(l)}, d_1^{(r)}, g_{i(t0)})$, $SJC(d_v, d_2^{(r)}, g_{i(t0)+1})$ are satisfied. Consequently, 2 Split operations (on $d_1^{(r)}$, d_v) take place in the range $D(l(d_v), r(d_v))$, and additional 2 Split operations, triggered by the candidates $d_1^{(r)}$, $d_2^{(l)} = d_v$, result outside of this range (on $d_1^{(l)}$, $d_2^{(r)}$). At t_0+1, $SJC(d_3^{(l)}, d_v, g_{i(t0+1)})$, which results in an additional Split ($d_3^{(l)}$) outside of this range, triggered by the candidate $d_3^{(r)} = d_v$. Finally, at t_0+2, $SJC(l(d_v), r(d_v), d_v)$ is fulfilled, triggering a Split operation on $l(d_v), r(d_v)$. In total, 7 Split operations result from the set $\{g_{i(t0)}, g_{i(t0)+1}, g_{i(t0)+2}\}$

(Observation 4), a Split operation on $d \in D_{v,j}(t_0)$ is triggered by *SC2* only if $d = \max(D_{v,j}(t_0))$. According to Observation 12 and Observation 13, the same can be argued regarding *SJC*. Therefore, a Split operation can be applied only on $d = \max(D[g_{j(t0)-1}, g_{j(t0)}])$ for any $g_{j(t0)-1}, g_{j(t0)} \in \Delta_v$. Moreover, let $g_{p(t0)}$, $g_{q(t0)}$ be the leftmost and rightmost inter-cluster gaps in the set Δ_v at iteration t_0, respectively. A Split operation will not take place in the ranges $D(l(d_v), g_{p(t0)})$, $D(g_{q(t0)}, r(d_v))$, since $l(d_v) = \max(D[l(d_v), g_{p(t0)})]$ and $r(d_v) = \max(D(g_{q(t0)}), r(d_v)])$. Hence, no more than $|\Delta_v| - 1$ Split operations will result from the set Δ_v in the range $D(l(d_v), r(d_v))$.

Lemma 9 The maximal number of Split operations triggered by $SJC(d^{(l)}, d^{(r)}, g_{i(t)})$, where $g_{i(t)} \in D(l(d_v), r(d_v))$ and $d^{(l)}$ or $d^{(r)} \notin D[l(d_v), r(d_v)]$, is $|\Delta_v|$.

Proof First we would like to show that if $SJC(d^{(l)}, d^{(r)}, g_{i(t)})$ is satisfied, where $g_{i(t)} \in D(l(d_v), r(d_v))$ then either $d^{(l)} \in D(l(d_v), r(d_v))$ or $d^{(r)} \in D(l(d_v), r(d_v))$ holds. Assume the opposite; that is, $D(l(d_v), r(d_v)) \subset D(d^{(l)}, d^{(r)})$. Since: (i) by definition $F(d^{(l)}, d^{(r)}) = true$, and (ii) following Observation 1, $\min(d^{(l)}, d^{(r)}) > \min(l(d_v), r(d_v))$, then $\min(l(d_v), r(d_v)) \in A_1$, which contradicts the definition of Δ_v when $l(d_v), r(d_v) \notin A_1$.

Thus, the only scenario where the Split operation resulting from Δ_v will take place on $d^{(l)} \notin D[l(d_v), r(d_v)]$ is when both $d^{(r)}, g_{i(t)} \in D(l(d_v), r(d_v))$. In cases when $d^{(r)} \in D(l(d_v), r(d_v))$ and $d^{(l)} \notin D[l(d_v), r(d_v)]$, we denote $d^{(r)}$ by $d^{(r)*}$. In the symmetric case when $d^{(l)} \in D(l(d_v), r(d_v))$ and $d^{(r)} \notin D[l(d_v), r(d_v)]$, we denote $d^{(l)}$ by $d^{(l)*}$. As demonstrated in Lemma 8, there are only $|\Delta_v| - 1$ inter-distances in the range $D(l(d_v), r(d_v))$ that can play the role of $d^{(r)*}$ (or $d^{(l)*}$) since $d^{(r)*} = \max(D[g_{j(t0)-1}, g_{j(t0)}])$ for $g_{j(t0)-1}, g_{j(t0)} \in \Delta_v$. To conclude, notice that any of those $|\Delta_v| - 1$ inter-distances can play the role of $d^{(r)*}$ (or $d^{(l)*}$) only once. This happens because if $SJC(d^{(l)}, d^{(r)*}, g_{i(t)})$ is satisfied, then $g_{i(t)}$ is the leftmost inter-cluster gap in the range $D(l(d_v), r(d_v))$. After this operation, $d^{(r)*}$ become the leftmost inter-cluster gap in this range, and therefore, will not play the role as $d^{(r)*}$ again. Following the same reasoning, only the last inter-distance removed from the $|\Delta_v| - 1$ Split candidates can play the role of both $d^{(r)*}$ and $d^{(l)*}$. This is because once inter-distance plays the role of $d^{(r)*}$ it can play the role of $d^{(l)*}$ only after the rest of the Split candidates have been classified as inner gaps.

Figure 11 illustrates the worst case scenario (regarding the number of Split operations) resulting from the set $\{g_{i(t0)}, g_{i(t0)+1}, g_{i(t0)+2}\}$.

Corollary 3 $|\Delta \cup \Psi| \leq 3.5 \cdot |\Delta|$.

Theorem 4 From any given starting point, the D-CUT algorithm converges to *GNOCA* under the assumption of stable configuration status. The convergence process requires $O(|\Delta|)$ worst case time and $O(\log|\Delta|)$ expected time, under the assumption of random permutation of the size of the inter-distances in the set $D(d_s, d_f)$.

Proof According to Corollary 2 the D-CUT algorithm converges to the *GNOCA* after no more than t_2 iterations. Notice that the refined height of the $SBT(D(d_s, d_f))$ is equal to the refined height of $SBT(\Delta \cup \Psi)$, since by definition this function only counts inter-distances that are classified as inner gaps. Furthermore, from Corollary 3: $|\Delta \cup \Psi| \leq 3.5 \cdot |\Delta|$. Hence, in the worst case the inter-distances in the set $\Delta \cup \Psi$ are organized in increasing/decreasing length order, and the convergence process requires $O(|\Delta|)$ iterations. The theorem follows since, according to Corollary 1, under the assumption of random permutation, $SBT(D(d_s, d_f))$ is a Random Binary Search Tree with expected height of $O(\log|\Delta|)$.

6 Simulation

In order to evaluate the performance of the D-CUT algorithm under realistic road conditions, we performed the following simulations.

6.1 Simulation setup

The D-CUT algorithm strongly depends on the inter-distances between vehicles. Thus, for faithful evaluation of the algorithm, a realistic mobility model for individual vehicles is

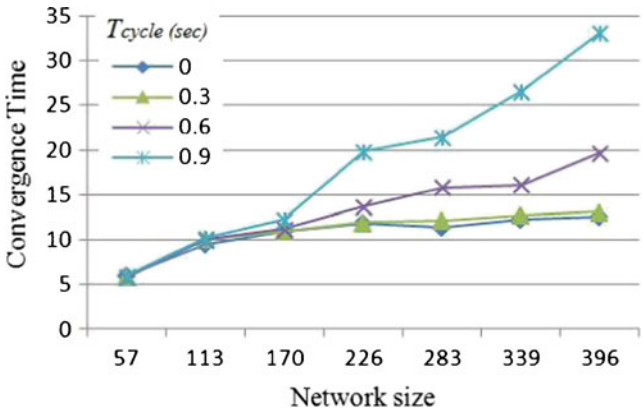

Fig. 12 Initial convergence time

required. Hence, we base our simulation on the microscopic model developed by Stefan Krauß [17] designed for multi-lane traffic flow dynamics. In this model, every vehicle has its own preferred speed, which the vehicle tries to reach if the conditions are satisfied (e.g., having enough safe distance). We set 20 % of the vehicles with 25 m/s preferred speed, 50 % with 35 m/s preferred speed, and 30 % with 40 m/s preferred speed. The vehicles with 25 m/s preferred speed correspond to the slow vehicles like heavy trucks; the rest of the drivers adjust their speed around the road speed limits (in this case 35—40 m/s). This situation corresponds to the dynamic changing flow on the road. In our highway traffic model, we assume that the vehicles run along a three-lane circular loop with a perimeter of 2,000 [m] and we consider traffic densities of 9, 18, 27, 36, 45, 54, and 63 vehicles per km. If not specified otherwise, the D-CUT free parameters are set as the following: R_{max} =250(meters), k_{max}=25, and p_{min}=2.

Since the D-CUT uses the vehicle proximity map provided by the beacon dissemination process as its input, the D-CUT cycle time needs to be in units of the beacon dissemination process cycle time. Here, we assume a 0.3 s beacon dissemination process cycle time and we consider the following D-CUT cycle times: 0.3, 0.6, 0.9, 1.2, 1.5, and 1.8 s.

6.2 Tracking the optimal solution

In our theoretical analysis we demonstrated a logarithmic convergence time under the assumption of a stable configuration. Here, our objective is to evaluate the convergence time during real traffic scenarios. First, we compare the initial convergence time of the clustering algorithms, and next, we compare their ability to track the optimal solution.

The initial convergence time is defined as the number of iterations required for the algorithm's convergence process, i.e., starting from an initial state in which each vehicle is considered a separate cluster, until the final state, in which no reorganizing-operation is applied by the clustering algorithm throughout an entire iteration. Figure 12 shows the

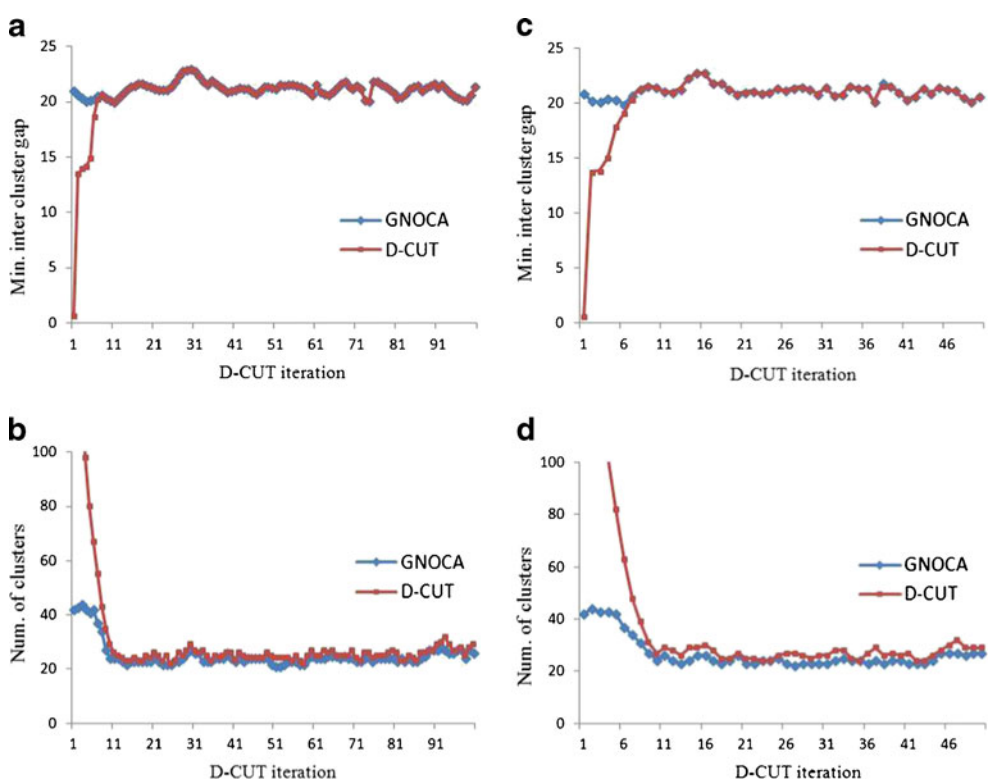

Fig. 13 Initial convergence process. **a** and **c** compare the minimal inter-cluster gap of the *CA* produced by the D-CUT algorithm with the minimal inter-cluster gap of *GNOCA* for time cycles of 0.3 s and 0.6 s, respectively. **b** and **d** compare the number of clusters of the two *CAs* for time cycles of 0.3 s and 0.6 s, respectively. In order to scale the process time, the first 100 iterations are plotted for 0.3 s cycle time and the first 50 iterations for the 0.6 s cycle time

Table 1 A comparison between the D-CUT *CA* and the *GNOCA* for each of the two objectives

Objective		D-CUT cycle time											
		0.3		0.6		0.9		1.2		1.5		1.8	
		1	2	1	2	1	2	1	2	1	2	1	2
Vehicle density	9	1.00	1.00	0.99	1.01	0.99	1.01	0.99	1.01	0.99	1.01	0.99	1.01
	18	1.00	1.01	0.99	1.02	0.99	1.03	0.99	1.04	0.99	1.05	0.99	1.05
	27	1.00	1.02	0.99	1.04	0.99	1.05	0.99	1.07	0.99	1.08	0.99	1.09
	36	1.00	1.04	0.99	1.07	0.99	1.10	0.99	1.12	0.99	1.14	0.99	1.15
	45	0.99	1.04	0.99	1.07	0.99	1.10	0.99	1.12	0.99	1.14	0.98	1.16
	54	1.00	1.04	0.99	1.08	0.99	1.11	0.99	1.14	0.99	1.15	0.99	1.17
	63	1.00	1.04	0.99	1.08	0.99	1.11	0.99	1.14	0.99	1.15	0.99	1.17

impact of network size on convergence time with different D-CUT cycle times (denoted by T_{cycle}). The zero cycle time corresponds to the stable configuration case. As we can see, the algorithm demonstrates a logarithmic convergence time for the different cycle times. The figure shows that for cycle times of 0.3 s the convergence time is generally the same as under a stable configuration. In medium and high density, a longer cycle time results in longer convergence time. In order to shed light on this effect, Fig. 13 presents the initial convergence process for 0.3 and 0.6 D-CUT cycle times. The figure compares the *CA* produced by the D-CUT algorithm with the *GNOCA* by comparing each of the two optimization objectives separately. Generally, this comparison shows that the D-CUT algorithm provides a fast convergence towards the optimal solution, and displays high correlation with it after initial convergence. In terms of Objective 1, in both cycle time settings, the D-CUT algorithm promptly converges to the *GNOCA* (around 8 iterations). The reason for the longer convergence time is seen when comparing the convergence in terms of Objective 2. When cycle time is set to 0.3 s the algorithm continues with the rapid convergence and completes the process at iteration 13. When cycle time is set to 0.6 s on the other hand, the D-CUT gets very close to the *GNOCA* at the same iteration, but due to the higher dynamic, fails to complete the process through additional 10 iterations.

After showing the initial rapid convergence, we continue by analyzing the ability of the D-CUT algorithm to maintain the *GNOCA*. In the following we use two different measures for evaluating this ability.

Table 1 presents the average ratio between the D-CUT *CA* and the *GNOCA* for each of the two optimization objectives with different D-CUT cycle times and vehicle densities. In more detail, at the end of each D-CUT iteration we evaluate the ratio between the minimum inter-cluster gap and the number of clusters of the two *CA*s. As we demonstrated in the previous section, the D-CUT converges to the *GNOCA* in two phases. First, the D-CUT replaces small inter-cluster gaps by larger ones up to the convergence to an optimal *CA* in terms of Objective 1. In the second phase, the algorithm greedily joins adjacent clusters, up to convergence to the *GNOCA*. Accordingly, as we can learn from this table, the algorithm obtains a ratio of nearly one in the context of Objective 1 under the different densities and even at cycle times as long as 1.8 s. In low density, the number of clusters ratio between the two *CAs* is nearly one as well. In medium and high densities, a high rate of D-CUT execution results in

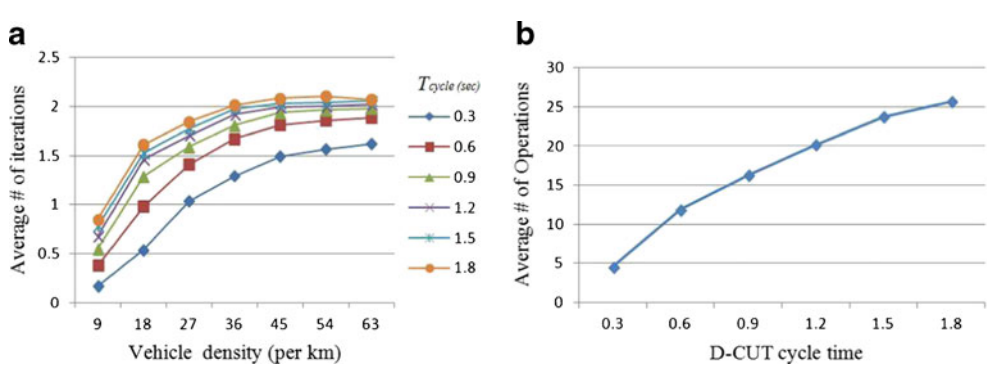

Fig. 14 **a** Average number of D-CUT iterations between the D-CUT *CA* and the *GNOCA* with different D-CUT cycle times. **b** Average number of D-CUT operations against cycle time for density of 63 vehicles per km

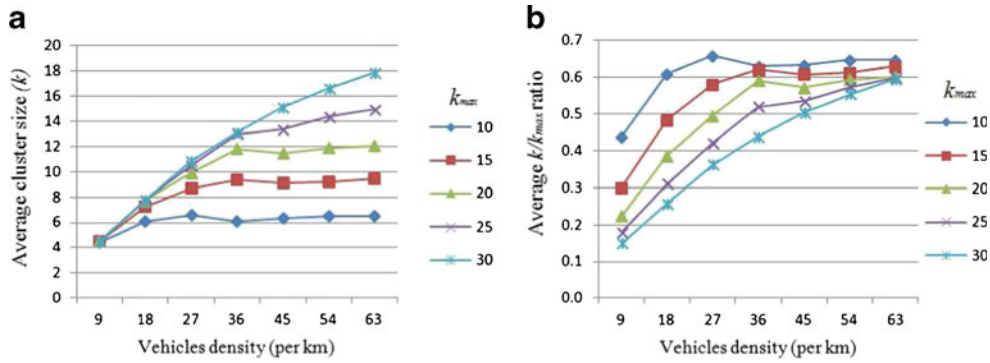

Fig. 15 **a** Average cluster size versus vehicle density with different k_{max} settings. **b** Average cluster size normalized by k_{max} versus vehicle density with different k_{max} settings

ratio of 1.04. The ratio increases with the increasing of cycle time and reaches 1.17 when the cycle time is set to 1.8 s.

The second measure we use to evaluate the ability of the D-CUT to maintain the *GNOCA* is presented in Fig. 14a. This figure shows the average number of D-CUT iterations standing between the D-CUT *CA* and the *GNOCA* versus the vehicle density with different cycle times. Specifically, at the end of each D-CUT iteration we freeze the configuration and evaluate the additional number of iterations required to complete the convergence process. From this figure we observe that the D-CUT algorithm is capable of following the *GNOCA* and keeping it within 2 iterations even for relatively high D-CUT cycle time (1.8 s) and under high density (63 vehicles per km).

An interesting point arising from this figure is that the increasing of D-CUT cycle time has a small benefit to this measure. For example, in a density of 63 vehicles per km, the difference between 0.9 and 1.8 cycle times is negligible (1 %). So even though the algorithm has updated the clustering at twice the rate, the results are almost the same. This observation can be explained by the fact that at a high execution rate, the algorithm is exposed to the configuration changes gradually, and thus, each D-CUT iteration is less effective since the algorithm cannot parallel its operation in an optimal way. This explanation is supported by Fig. 14b that shows the average number of D-CUT operations against the D-CUT cycle time at a density of 63 vehicles per km. As we can see, the average number of operations increases with the increase in the D-CUT cycle time.

The aforementioned finding implies an efficient clustering strategy in which the D-CUT is executed at a low rate (e.g., $T_{cycle} = 1.8$). To avoid the 17 % growth in the number of clusters, every few seconds the D-CUT will initiate a rebuilding process (that will take on average two iterations) in order to converge to the *GNOCA*.

6.2.1 Cluster size and channel utilization

In the following we want to evaluate the influence of the free parameter k_{max} on the D-CUT performance. As we said earlier, we limit the cluster size by k_{max} in order to guarantee channel allocation for each cluster member. Thus, the ratio between the average cluster size and k_{max} is a good indication for the bandwidth utilization during the beacon dissemination process.

In our objective function, there are two criteria defining cluster validity: the first is the existence of p_{min} clusterhead candidates, which cover the entire cluster population within their transmission range. The second is limiting the cluster size by k_{max}. Figure 15 presents the effect of this limitation on the average cluster size. Figure 15a presents the average cluster size versus vehicle density with different k_{max} settings, and Fig. 15b presents the same but with average cluster size normalized by k_{max}. In low density, cluster size is determined according to the first criterion, and thus, the k_{max} criterion has no effect. However, when density increases valid clusters become mainly dependent on the k_{max} criterion. In high

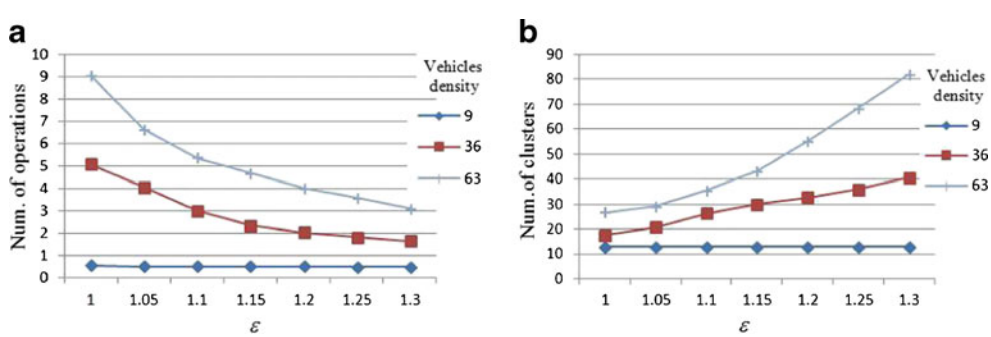

Fig. 16 D-CUT relaxed model performances. **a** The number of D-CUT operations against the trade-off balancing free parameter ε with low, medium, and high densities. **b** The effect of ε on the number of clusters in the model for the same densities

density, the average cluster size is about 60–65% of k_{max}. The figure highlights the scalability of the algorithm. This is seen by better bandwidth utilization when vehicle density, and thus the channel load, increase.

Notice that the second validity criterion poses a lower bound of n/k_{max} clusters for every valid *CA*. As we can see from the dynamic simulation, the ratio achieved is even better than the ratio of 3 proved before for static configuration.

6.2.2 The relaxed D-CUT

In this section we introduce a relaxed form of our D-CUT algorithm designed to refine the trade-off between two contradicting objectives, stability and adaptivity. The traditional D-CUT algorithm greedily replaces small inter-cluster gaps by larger gaps. To slow down this process, the relaxed D-CUT algorithm triggers those replacements only when the difference between the two gaps is substantial. In particular, the replacement will take place only if the ratio between the gaps is larger than a predefined ε, where ε is the trade-off balancing free parameter.

Figure 16 presents the influence of this free parameter on the algorithm performance. Figure 16a shows the average number of D-CUT operations throughout a D-CUT iteration, and Fig. 16b shows how this affects the number of clusters produced by the algorithm (the effect on minimal inter-cluster gap is small and thus is not presented). The figure presents the results for low density (9 vehicles per km), medium density (36 vehicles per km), and high density (63 vehicles per km). In low density, the trade-off balancing free parameter has no effect either on the (anyway low) number of operations or on the number of clusters produced by the algorithm. In medium and high density, the figures show that the increase of ε is a result of the increase of stability at the direct expense of the clustering adaptivity.

7 Conclusions

In this paper we present the D-CUT algorithm designed specifically to provide extensive but reliable inter-cluster bandwidth reuse. To this end, the D-CUT algorithm performs under the following objectives: 1) the algorithm seeks to minimize the inter-cluster interference by producing clusters which are separated by the maximal possible inter-cluster gaps and 2) the algorithm aims to increase the cluster to its maximal size; thereby allowing the most efficient utilization of the allocated bandwidth. By extensive theoretical analysis we have demonstrated a coordinated, local and fast convergent algorithm that produces Geographically Near-Optimal Clustering Assignment (*GNOCA*) from any initial clustering assignment. We prove logarithmic convergence time based solely on the distances between the initial clustering assignment and the *GNOCA*. We also performed simulation analyses in order to evaluate the performance of the D-CUT algorithm under realistic road conditions. Our simulation results support our theoretical findings with respect to logarithmic initial convergence time under realistic traffic scenarios. Our simulation results also show the capability of the algorithm to follow the constantly changing *GNOCA*. It is also demonstrated that the D-CUT algorithm keeps the *GNOCA* within two iterations with very close bounds compared to the optimal values.

Appendix 1—list of abbreviations

CA	Clustering Assignment
D-CUT	Distributed Construct Underlying Topology
GNOCA	Geographically Near-Optimal *CA*
JC1	the first join condition
JC2	the second join condition
MMIDP	Max-Min Inter-Distance Pair
SC1	the first split condition
SC2	the second split condition
SJC	the spit join condition
VANET	Vehicular ad-hoc network

Appendix 2—an example of the D-CUT convergence process

In the following we provide a detailed example of the D-CUT convergence process.

The D-CUT gets as an input 9 small scrappy clusters (see Fig. 17a). Next we show how D-CUT converges to the *GNOCA* in three iterations that match the height of the corresponding *SBT*.

Throughout the convergence process 14 inter-distances are involved. Those inter-distances are associated to the class A_1, A_2, and A_3 as the following: $\{d_1, d_2, d_3, d_4, d_5, d_6, d_8, d_9, d_{10}, d_{12}, d_{14}\} \in A_1$; $\{d_{13}\} \in A_2$; $\{d_7, d_{11}\} \in A_3$.

In first iteration (see Fig. 17b) the inter-distances $\{d_1, d_3, d_5, d_8, d_{10}, d_{12}, d_{14}\}$ (which belong to A_1) satisfy the *SJC* and are classified as inner-gap (Lemma 2). In addition, the inter-distance d_{13} that belong to A_2 is classified as an inter-cluster gap (Lemma 4).

In the second iteration (see Fig. 17c) the inter-distances $\{d_2, d_6, d_9\}$ that belong to A_1 satisfy the *SJC* and are classified as inner-gap. At the end of this iteration (after all inter-distances that belong to A_1 are classified as inner-gap), the *CA* produced by the D-CUT algorithm meets Objective 1 (Lemma 3).

In the third iteration (see Fig. 17d), the inter-distance d_{11} that belongs to A_3 satisfies the *JC1* and is classified as an inner-gap. In addition, the inter-distance d_7 that belongs to A_3

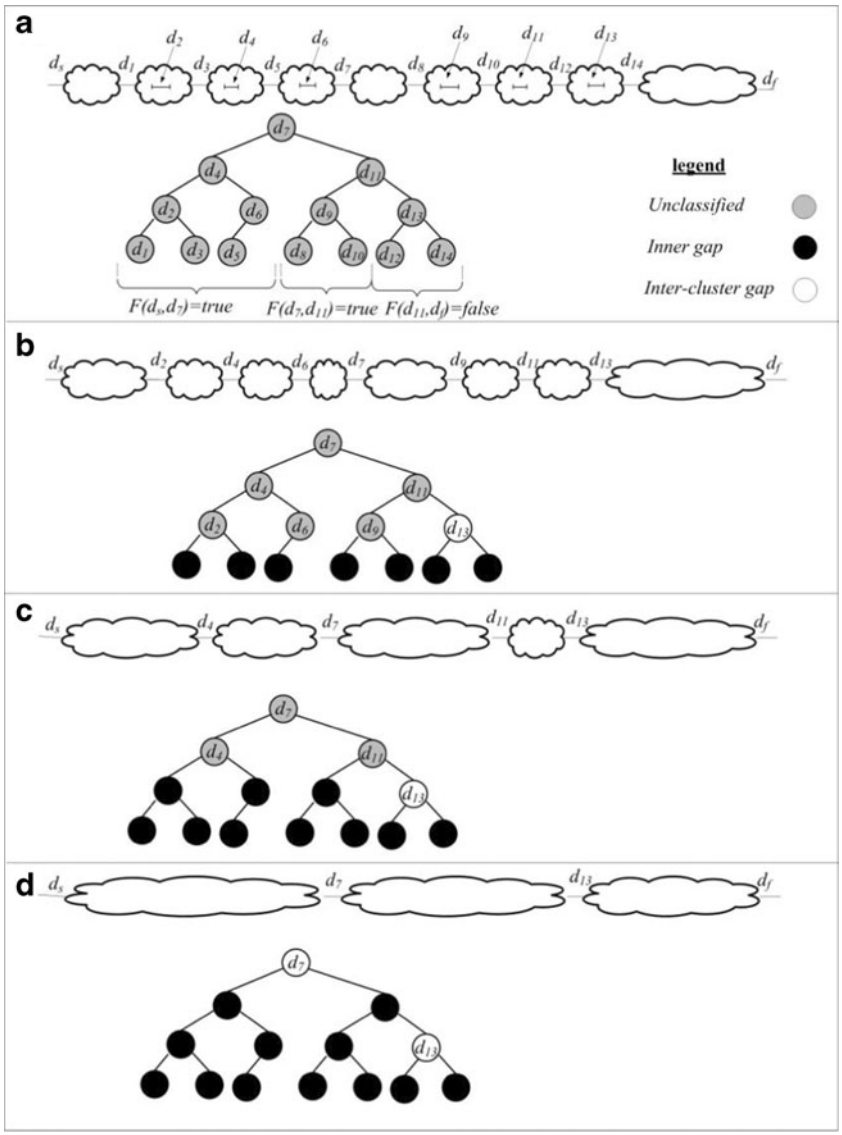

Fig. 17 An example of D-CUT convergence process

is classified as an inter-cluster gap (since $F(d_7, d_f)=false$; see Lemma 5). At the end of this iteration the D-CUT algorithm produces the *GNOCA* (Lemma 6).

References

1. Kumar D, Kherani A, Altman E (2006) Route lifetime based optimal hop selection in VANETs on highway: an analytical viewpoint. In: Proceedings of IFIP Networking, Coimbra, Portugal
2. Baldessari R et al. (2007) Car-2-car communication consortium-manifesto. In: DLR Electronic Library [http://elib.dlr.de/perl/oai2] (Germany)
3. Torrent-Moreno M, Santi P, Hartenstein H (2005) Fair sharing of bandwidth in VANETs. In: Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks, pp. 49–58
4. Gunter Y, Wiegel B, Großmann HP (2007) Cluster-based medium access scheme for vanets. In: IEEE Intelligent Transportation Systems Conference, pp. 343–348
5. Su H, Zhang X (2007) Clustering-based multichannel MAC protocols for QoS provisionings over vehicular ad hoc networks. IEEE Trans Veh Technol 56(6):3309–3323
6. Tian D, Wang Y, Lu G, Yu G (2010) A VANETs routing algorithm based on Euclidean distance clustering. In: 2nd International Conference on Future Computer and Communication, vol. 1, pp. 183–187
7. Wischhof L, Ebner A, Rohling H (2005) Information dissemination in self-organizing intervehicle networks. IEEE Trans Intell Transp Syst 6(1):90–101
8. Bononi, Di Felice M (2007) A cross layered mac and clustering scheme for efficient broadcast in vanets. In: IEEE International Conference on Mobile Adhoc and Sensor Systems, pp. 1–8
9. Fan P (2007) Improving broadcasting performance by clustering with stability for inter-vehicle communication. In: IEEE 65th Vehicular Technology Conference, pp. 2491–2495
10. Raya M, Aziz A, Hubaux JP (2006) Efficient secure aggregation in VANETs. In: The 3rd International Workshop on Vehicular ad hoc Networks, pp. 67–75
11. Fan P, Haran J, Dillenburg J, Nelson P (2005) Cluster-based framework in vehicular ad-hoc networks. In: Ad-hoc, mobile, and wireless networks, pp. 32–42

12. Wang Z, Liu L, Zhou MC, Ansari N (2008) A position-based clustering technique for ad hoc intervehicle communication. IEEE Trans Syst Man Cybern Part C Appl Rev 38(2):201–208
13. Kenichi M, Yoshiyuki W, Nobuhito M, Nakano K, Sengoku M (2002) Flooding schemes for clustered ad hoc networks. IEICE Trans Commun 85(3):605–613
14. Kayis O, Acarman T (2007) Clustering formation for inter-vehicle communication. In: IEEE Intelligent Transportation Systems Conference, pp. 636–641
15. Papadimitratos P, Buttyan L, Holczer T, Schoch E, Freudiger J, Raya M, Ma Zhendong Z, Kargl F, Kung A, Hubaux JP (2008) Secure vehicular communication systems: design and architecture. In: IEEE Communications Magazine 46(11): 100–109
16. Reed B (2003) The height of a random binary search tree. J ACM 50(3):306–332
17. Krauß S, Wagner P, Gawron C (1997) Metastable states in a microscopic model of traffic flow. Phys Rev E 55:55–97