Adaptation Strategies for Self-management of Tree Overlay Networks

Evangelos Pournaras Martijn Warnier Frances M.T. Brazier
Systems Engineering Section
Department of Multi-actor Systems
Delft University of Technology
Delft, The Netherlands
{*E.Pournaras,M.E.Warnier,F.M.Brazier*}*@tudelft.nl*

***Abstract*—Self-management of tree overlay networks for distributed applications is the challenge this paper addresses. Eight local adaptation strategies are introduced based on which autonomous self-organized agents establish connections that build and maintain a tree topology. Quantitative and qualitative experimental evaluation illustrates and compares the effects of adaptation strategies in the resulting tree topologies according to a defined self-organization goal and four metrics: connectedness, connectivity, instability and robustness. This paper concludes that further applicability of adaptation strategies in other self-organization goals and topologies is promising.**

***Keywords*-self-management; self-organization; adaptation; strategy; tree overlay; agent;**

I. Introduction

A wide range of distributed applications rely on hierarchical virtual networks for communication between agents. These virtual networks, defined by overlay topologies, provide the basis for operations such as aggregation, information dissemination, decision-making etc. Various distributed applications are based on these operations, such as application-level multicasting [1], energy self-management [2], [3] and distributed database management [4]. Tree topologies are designed to optimize such applications. For example, a broadcast tree overlay for multimedia multicasting is designed to minimize latency whereas, a tree structure in a distributed database is designed to offer data redundancy.

In this paper, self-management refers to the emergence of global qualities in a tree topology by using self-organized agents with different behaviors. Self-management of tree overlays influences performance of distributed applications [5], [6]. For example, in application-level multicasting, a 'short'[1] self-organized tree is acquired with agents sorted based on one or more pre-defined criteria [1], e.g. bandwidth, agent availability etc. More robust agents are placed near the root of the tree with the less robust positioned further down in the tree.

Defining and designing local agent behaviors that collectively emerge such complex tree topologies is challenging. This paper proposes eight adaptation strategies that represent organization preferences of agents. Adaptation strategies control the parent and children selected by the agents of a tree topology. When an agent applies these local strategies, different tree topologies emerge. These topologies and their effects are evaluated qualitatively and quantitatively based on four performance metrics and a defined self-organization goal.

The contributions of this paper are outlined as follows:

- The self-organization goal and its relation to a tree overlay and its properties (Section III).
- Introduction of four performance metrics based on which the self-organization goal is evaluated when different adaptation strategies are adopted (Section III).
- Introduction of eight adaptation strategies based on which an agent selects its parent and children. This is the core contribution of this paper (Section IV).
- Qualitative and quantitative evaluation of adaptation strategies in various simulation scenarios (Section V).

Adaptation strategies provide a flexibility to agents to adopt different connection preferences. Unlike existing self-organization systems that are based on a single static ranking function [7], [8], [9], [10], adaptation strategies have different effects in the resulting self-organized tree topologies. Based on these effects, performance trade-offs can be explored regarding different application requirements and domains.

II. Problem Description

This paper focuses on the problem of self-managing tree overlay networks for distributed applications. This paper shows that self-management of tree topologies emerges from a collective behavior of self-organized agents. Defining and designing eight *adaptation strategies* for achieving this collective behavior is the goal of this paper.

Every agent in a network is ranked according to a criterion defined by a distributed application. The criterion can be related with the availability of agent, its reliability, trust, bandwidth etc. An agent is ranked based on the value of such a criterion. An agent is part of a tree overlay by establishing connections with a *parent* agent and *children* agents. From an agent point of view, the problem is which ranked parent and children an agent should connect with to maximize

[1]In a 'short' tree, agents are connected with a high number of children.

the performance of an application according to the chosen criterion.

This paper defines four performance metrics and the properties of a tree topology in which agents aim to self-organize themselves. The effects of adaptation strategies in self-organization are evaluated. Conclusions can be drawn about the effects of different collective behaviors in self-organized tree topologies. Based on these effects, performance tradeoffs can be explored regarding different application requirements.

III. Self-organization Goal

Assume a network of n agents. Each agent i has a rank value r_i in the range $[0, 1)$. This rank is assigned by the application and represents a chosen performance criterion. Assume also that each agent has an agent degree d. Therefore, each agent can support a maximum of $k = d-1$ children and 1 parent. Each agent acquires a higher ranked agent for its parent and lower ranked agents for its children.

The goal of self-organization is the formation of a balanced k-ary tree that (i) is fully connected (one and only one tree) (ii) is sorted according to the rank of agents and (iii) agents are connected with a maximum number of other agents $d = k + 1$ except for the leaves ($d - k = 1$) and the root ($d - 1 = k$).

Next assume that given an initial state with n disconnected agents, every agent selects its parent and children according to a strategy j. Note that for now, a strategy represents which rank values are preferred. A detailed illustration is provided in Section IV. Tree topologies are evaluated based on convergence of the following performance metrics: (i) *connectedness*, (ii) *connectivity* (iii) *instability* and (iv) *robustness*.

Connectedness is the proportion of agents connected to the tree with the highest number of agents (the main tree). The connectedness $\hat{c}$ of a strategy j for a forest $\boldsymbol{F}$ of m trees $\{\boldsymbol{T}_0, \boldsymbol{T}_1, ..., \boldsymbol{T}_{m-1}\} = \boldsymbol{F}$ is defined as follows:

$$\hat{c}(j) = \frac{\max(|\boldsymbol{T}_0|, |\boldsymbol{T}_1|, ..., |\boldsymbol{T}_{m-1}|)}{n} \tag{1}$$

In contrast, the tree connectivity for a strategy j is defined as the number of connections established in the agents in relation to the maximum number of connections that can be supported, that is the agent degree d. Connectivity is interpreted as the capability of a strategy to create the maximum number of links required for the formation of the optimum topology. It is calculated by dividing the actual connectivity $\overline{c}(j)$ achieved by the strategy j over the optimum connectivity c' of an optimum k-ary tree as defined in this section:

$$c(j) = \frac{\overline{c}(j)}{c'} \tag{2}$$

The actual connectivity $\overline{c}$ of a strategy j is the average connectivity of all agents. This is the number of parents $x = 0|1$ and children $y \leq k$ of every agent i in relation to the agent degree d such that:

$$\overline{c}(j) = \frac{\sum_{i=0}^{n-1}(x_i + y_i)}{n * d} \tag{3}$$

Similarly, the optimum connectivity c' is the average connectivity of all agents in an optimum k-ary tree. It is calculated by determining the aggregate result of the maximum agent connectivity for all the agents $n * d$ and subtracting $d-1$ connections for each leaf and 1 connection for the root. The upper bound for the number of leaves l is calculated as follows [11]:

$$l = k^{\log_k(k-1) + \log_k n - 1} \tag{4}$$

Based on the knowledge about the number of leaves in the optimum tree, the optimum connectivity can be summarized as:

$$c' = \frac{n * d - l * (d-1) - 1}{n * d} \tag{5}$$

Instability is another metric introduced in this paper for evaluation of self-organization in tree topologies. Instability indicates numerically how much connectivity changes during convergence. High instability for a strategy j means that this strategy imposes many connection changes before finding the best children and parent for each agent. In contrast, low instability does not result in frequent changes in the parent and children connections of agents. Low instability does not necessarily imply a better strategy. This is because a strategy 'trapped' in a local optimum is very stable but not flexible enough to 'jump' to a more optimum connection [12].

The instability s for a strategy j is calculated by computing the average standard deviation $s_i(j)$ of agent connectivities over a period of time T such that:

$$s(j) = \frac{\sum_{i=0}^{n-1} s_i(j)}{n} \tag{6}$$

where:

$$s_i(j) = \sqrt{\frac{1}{T-1}\sum_{t=0}^{T-1}[c_i^t(j) - \overline{c}_i(j)]^2} \tag{7}$$

Note that, the instability $s_i(j)$ for an agent i and a strategy j is calculated using the classic statistics formula of standard deviation. The $\overline{c}_i(j)$ is the average connectivity of an agent i and a strategy j over a time period T. The $c_i^t(j)$ is the connectivity of agent i at time t using strategy j. The final instability in the tree built with strategy j is calculated by averaging the instability values of all the agents as illustrated in Formula (6).

Finally, the robustness q of the tree expresses the quality of the sorting process and agent connections. It is associated with the ranking fitness of agents with their neighbors. This paper correlates the robustness q with the the *relative*

ranking distance between the agents and their parents. This is the distance $r_p - r_i$ that an agent i has from its parent p in relation to the maximum distance that can have from the highest ranked agent in network. Assuming agents with uniformly distributed ranks, the maximum possible ranking distance is approximately $1 - r_i$. Therefore, the robustness q for a strategy j is defined as:

$$q(j) = \frac{\sum_{i=0}^{n-1} q_i(j)}{n} \tag{8}$$

where,

$$q_i(j) \approx \frac{r_p - r_i}{1 - r_i} \tag{9}$$

Although there are many graph configurations that create a balanced and connected k-ary tree, there is one and only one sorted tree that maximizes the robustness, presented by Formula (8), given the constraints of this problem.

IV. Topology Self-management

This section illustrates how an agent selects to connect to its 'best', according to a strategy, ranked parent and children. In contrast to a single static preference scheme, pre-defined from a fitness function, this paper shows that an agent has the option to adopt different ranking preferences for selecting its parent and children. These different preferences form the adaptation strategies that are able to self-organize agents into different tree topologies.

The goal of agents is to self-organize themselves in a sorted tree topology according to a criterion defined by an application, as mentioned in Section III. Each agent in the network performs the following feedback loop of five tasks:

1) Collect information about other agents in the network and form its views.
2) Select agents from views that are in close (rank) proximity according to an adaptation strategy. Sort the selected ranked agents in a list: the search space for parent and children
3) Select a parent and children from the search space according to an adaptation strategy.
4) Request connections from the selected parent and children.
5) Receive an acknowledgement or rejection for each connection. Based on this outcome:
    a) Establish new connections and removes older ones.
    b) Adapt the agent's search space to improve future selections of a parent and children.

Figures 1 outlines the local feedback loop of the five tasks executed by an agent. Based on this loop, agents are self-organized in a sorted tree topology as defined in Section III. These steps are executed periodically and the time period is a parameter of controlling performance, processing and communication overhead.

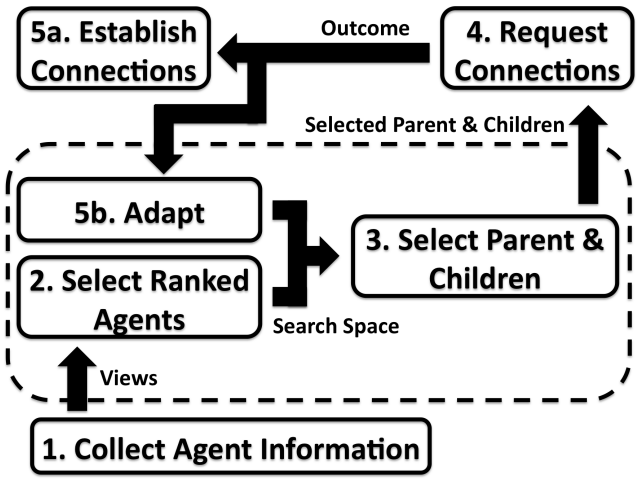


Figure 1. Local execution of five agent tasks. The dotted line encloses the tasks controlled by adaptation strategies.

A. Determining the Views

An agent discovers information about other agents in network by performing gossiping information exchange [9], [13]. This information is the views of an agent and concerns the contact address of other agents and their rank values according to the chosen criterion. The view has a limited size and is thus not global. An agent gossips periodically and updates its views with information about other agents. Collected agents can be random [13] or in close (rank) proximity [9] according to the criterion. A random view guarantees that the network remains fully connected and is resilient to agent failures. A proximity view clusters agents according to their ranking distance[2]. By keeping agents in close proximity, searching with the goal of self-organization becomes more efficient.

B. Determining the Search Space

An agent selects its parent and children from a search space. This is a sorted list of agents received from views. The search space is adaptive as constraints are applied in the ranges of rank values that are allowed to be present in this sorted list. For example, assume that every agent i has a unique rank r_i in the range $[0, 1)$. Agent i adapts its search space to contain ranked agents in the range $[0, r_i) \cup (r_i, 1)$. Therefore, the constraint in this adaptation is the exclusion of r_i from the search space.

Adaptations enable agents to perform selections of their parent and children within a range of ranks that improves system performance. In this way, convergence can be achieved to the sorted tree topology with the desired properties as defined in Section III. Section IV-E provides

[2]The ranking distance between two agents i and j refers to their euclidean distance $|r_i - r_j|$.

examples and explains in detail the applied adaptations in the search space.

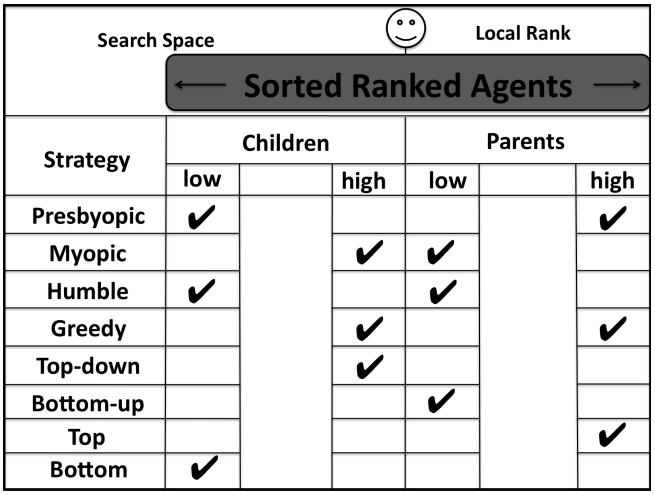
C. Adaptation Strategies

An agent connects with its parent and children based on one of the eight strategies depicted in Figure 2 and illustrated below:

- **Presbyopic**: An agent selects the highest ranked parent and the lowest ranked children from its search space.
- **Myopic**: An agent selects the lowest ranked parent and the highest ranked children from its search space. The defined preferences of the 'myopic' strategy are the opposite of the 'presbyopic' strategy.
- **Humble**: An agent selects the lowest ranked parent and the lowest ranked children from its search space.
- **Greedy**: An agent selects the highest ranked parent and the highest ranked children from its search space. Note that, these preferences are the opposite ones of the 'humble' strategy.
- **Top-down**: An agent selects the highest ranked children from its search space. Selection of a parent is not performed. The parent connection is established from received requests.
- **Bottom-up**: An agent selects the lowest ranked parent from its search space. Selection of children is not performed. The children connections are established from received requests.
- **Top**: An agent selects the highest ranked parent from its search space. Selection of children is not performed and children connections are established from received requests. Note that, the preference of this strategy is the opposite of 'bottom-up'.
- **Bottom**: An agent selects the lowest ranked children from its search space. Selection of a parent is not performed and a parent connection is established from received requests. Note that, the preference of this strategy is the opposite of 'top-down'.

The eight adaptation strategies represent the eight, in total, combinations of preferences that agents can have for high or low ranked children and/or parents. This is explained as follows: An adaptation strategy represents the preference to high ranked agents or low ranked agents (2 options). Furthermore, an agent may prefer to select (i) only higher ranked agents than itself (parent), (ii) only lower ranked agents than itself (children) or (iii) both. This provide 3 more options in the selection scheme of agents. Therefore, the total number of preference combinations that an agent can have for selecting its parent and children are $2^3 = 8$. These eight combinations are the adaptation strategies of agents.

Adopting preferences to both low and high ranked agents plays a crucial role in the self-organized topologies. For example, assume that highly ranked agents represent robust agents based on a performance metric. Intuitively, a 'greedy' strategy can provide high robustness as all the agents tend to connect with the highest possible ranked agents. In contrast, the 'humble' strategy is based on low ranked parents and children. This does not mean necessarily that the formed topologies are not robust. From the local viewpoint of a 'humble' agent, a connection with a low ranked child is not robust. However, from the local viewpoint of this child, the connection is highly robust. Section V illustrates how these small differences in the local preferences are reflected in the emerged tree topologies.



Strategy	Children			Parents		
	low		high	low		high
Presbyopic	✔					✔
Myopic			✔	✔		
Humble	✔			✔		
Greedy			✔			✔
Top-down			✔			
Bottom-up				✔		
Top						✔
Bottom	✔					

Figure 2. The ranking preferences of agents for parents and children in each adaptation strategy.

Note that, the search space is filled with ranked agents according to the same preferences of the adopted adaptation strategy. This means that if an agent selects high ranked parents and children ('greedy' strategy), the proximity for filling the search space is also based on the highest ranking distance. Strategies do not only concern the selection preferences but also how the search space is adapted. More information about adaptations are provided in Section IV-E.

D. Configuration of Connections

Links in a tree topology are bidirectional. This means that a connection is acknowledged and established between the two agents that form a link in a tree. Therefore, when an agent selects a parent and children based on a strategy, it sends a request to each of them thereby acquiring a connection. When the parent or child receives the request, it checks if the new connection is 'better', according to its strategy, than a connection that the agent already has. Specifically, the parent or child evaluates if the rank of this agent matches its preferences for low ranked or high ranked agents. Note that, the number of connections established respect the agent degree d of agents. Therefore, a new connection may replace an existing one.

Assume the following example scenario: A ranked agent $r_i = 0.5$ is connected with a ranked agent $r_p = 0.7$ and

adopts the 'greedy' strategy. Agent i receives two requests from the ranked agents $r_a = 0.6$ and $r_b = 0.8$. The connection with the ranked agent $r_a = 0.6$ is rejected. The ranked agent $r_p = 0.7$ with which there is an established connection has a higher rank than r_a. Therefore, r_p is preferred according to the 'greedy' strategy. In contrast, the ranked agent $r_b = 0.8$ has higher rank than $r_p = 0.7$. In this case, the established connection with agent p is removed and a new one is established with agent b. Note that, an agent is allowed to have one and only one parent.

To sum up, a new improved connection is established either (i) when an agent selects a parent or child and a connection request sent to this selected agent is acknowledged, or (ii) when a connection request is received by an agent and the requested agent is ranked 'better' according to the adopted strategy.

E. Adaptations

An agent uses the outcome of its connection requests as a feedback mechanism to adapt its search space. The acknowledgment of a request is named a *positive feedback* for both agents that establish a parent-child link. The rejection of a request or a removal of a connection is named a *negative feedback* and it is applied to the requester agent and the removed parent or child respectively.

Based on this feedback, an agent triggers adaptations to its the search space. As mentioned in Section IV-B, adaptations change the range of ranked agents that can be present in the sorted list (search space) from which parents and children are selected. The new ranges of ranks in the search space change in reference to the selected parent or child that causes the positive or negative feedback. Assume a ranked agent $r_i = 0.5$ that adopts the 'greedy' strategy. Initially, as given in the example of Section IV-B, the range of ranks in the search space is $[0, r_i) \cup (r_i, 1)$.

Assume also that agent i selects the ranked agent $r_a = 0.7$ as parent. Agent a accepts the connection and this provides positive feedback to the search space of agent i. Agent i adapts the range of ranks in its search space to $[0, r_i) \cup (0.7, 1)$ given, for example, the 'greedy' strategy. This adaptation provides a constraint in the new selected parent to be ranked higher than 0.7. In this case, this improvement is defined by the 'greedy' strategy.

In contrast, negative feedback is applied after a rejection or removal of a connection. Assume that after the latter adaptation, agent i selects from the range $[0, r_i) \cup (0.7, 1)$ the ranked agent $r_b = 0.9$. Assume also that agent b has higher ranked children than the ranked agent $r_i = 0.5$. Agent b rejects the connection and therefore, agent i adapts its search space based on negative feedback. The new ranges of ranked agents are $[0, r_i) \cup (0.7, 0.9)$. This adds an additional constraint for improving the parent connection: A new selected parent should be higher than 0.7 but lower than 0.9.

Finally note that if the search space cannot provide new parents and children, the ranges are reset back to $[0, r_i) \cup (r_i, 1)$ and adaptations apply again.

V. Evaluation

This section illustrates the parametrization of simulation environment and the results from the evaluation of adaptation strategies.

A. Simulation Environment

The self-organization agent and its adaptation strategies are implemented and simulated in ProtoPeer [14], a toolkit for prototyping distributed systems. The tested network consists of 1500 agents. Every agent is assigned a random rank value in $[0, 1)$ derived from an abstract application criterion. An agent can connect with a maximum number of children $k = 4$. The system runs for up to 400 execution rounds, referred to as *epochs* in ProtoPeer terminology. During each epoch, the views of agents are updated 5 times using gossip information exchange [9], [13]. Moreover, the feedback loop of agent tasks, illustrated in Figure 1, is executed 2 times per epoch. Therefore, an agent sends 2 connection requests per epoch at maximum.

Eight tree topologies are simulated, one for each adaptation strategy. Agents adopt the same strategy for each simulated topology. The resulting topologies are evaluated quantitatively according to connectedness, connectivity, instability and robustness as illustrated in Section III. Qualitative comparison is also provided based on visualization[3] of the resulting tree topologies. Results are compared to the optimum values of the 'ideal tree' when this is possible. The 'ideal tree' is simulated based on a centralized tree building mechanism. The performance metrics of the 'ideal tree' are computed mathematically and based on simulations. Finally, the communication cost of the strategies is outlined by calculating the aggregate number of requests, acknowledgments, rejections and removals sent by all agents in network.

B. Results

The purpose of the evaluation of strategies is to show which strategy is the best for self-organization of the tree topologies illustrated in Section III and evaluate the effect of each strategy in the formed topology.

Figure 3 illustrates the convergence of connectedness for the eight strategies. 'Top-down' and 'myopic' in Figure 3a perform the best. The 'top-down' strategy approaches 100% connectedness. Note that convergence requires 50 epochs to approach the average maximum values. The 'bottom-up' and 'greedy' strategies in Figure 3b have 81.85% and 80.21% average connectedness respectively. However, note that the 'bottom-up' deviates 2.5% more than the 'greedy' strategy.

The 'humble' and 'presbyopic' strategies in Figure 3c have a lower average connectedness of 79.30% and 60.30%.

[3]Visualization is based on the JUNG library [15].

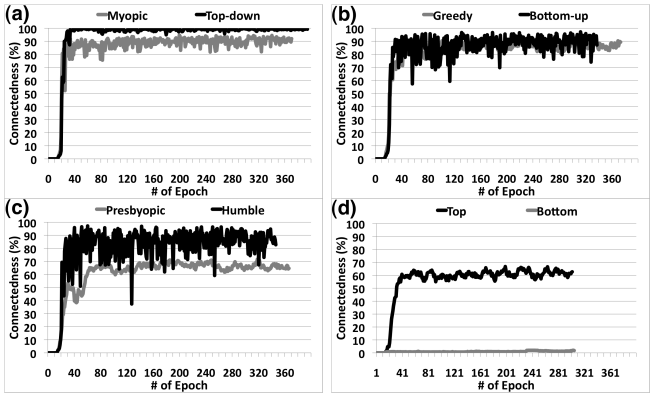


Figure 3. Convergence of connectedness for the eight adaptation strategies. (a) 'Top-down' and 'myopic'. (b) 'Bottom-up' and 'greedy'. (c) 'Humble' and 'presbyopic'. (d) 'Top' and 'bottom'.

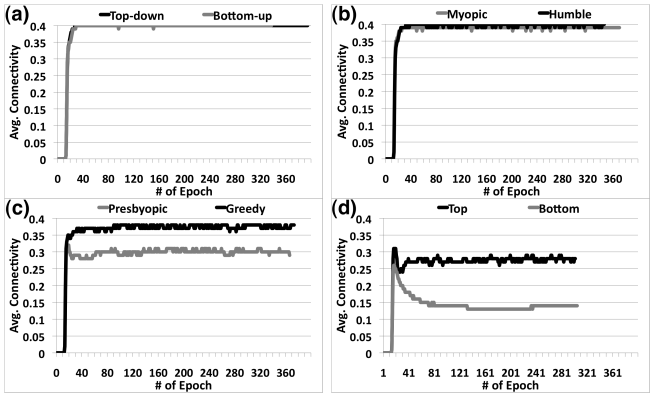


Figure 4. Convergence of average connectivity for the eight adaptation strategies. For these experimental settings, the optimum connectivity is 0.4. (a) 'Top-down' and 'bottom-up'. (b) 'Humble' and 'myopic'. (c) 'Greedy' and 'presbyopic'. (d) 'Top' and 'bottom'.

'Presbyopic' deviates 5.5% more than 'humble'. Finally, the 'top' and 'bottom' strategies perform the worst. 'Top' converges to an average connectedness of 55.99%. The 'bottom' strategy seems completely inappropriate with 0.99% average connectedness. Agents in the 'bottom' strategy try to connect with the lowest ranked agents that is possible. This concept is opposing to the self-organization goal and therefore agents cannot establish connections.

For the evaluation of connectivity, the optimum connectivity c' is calculated. First, the number of leaves must be counted. From Formula (4), the number of leaves are $l = k^{\log_k(k-1)+\log_k n-1} = 4^{5.068} \approx 1125$. Then, the optimum connectivity can be calculated according to Formula (5) as $c' = \frac{n*d-l*(d-1)-1}{n*d} = (1500*5-1125*4-1)/(1500*5) \approx 0.4$.

Figure 4 illustrates the convergence of the average connectivity $\bar{c}$ for the eight strategies. The 'top-down', 'bottom-up', 'myopic' and 'humble' strategies converge to the optimum connectivity as shown in Figure 4a and Figure 4b. The first two achieve faster convergence and fewer deviation from the optimum value of 0.4. 'Greedy' has an average connectivity of 0.36 over 400 epochs as shown in Figure 4c. At the same figure, 'presbyopic' follows with average connectivity of 0.29 during convergence time. Finally, the 'top' and 'bottom' strategies in Figure 4d perform the worst with 0.26 and 0.14 average connectivity during convergence time.

For the evaluation of tree robustness, the optimum value is measured by building and simulating the 'ideal tree'. The optimum robustness value is 0.74 in this case. Figure 5 illustrates the convergence of robustness for the eight adaptation strategies. Figure 5a shows that the 'top-down' strategy approaches the optimum value with the 'greedy' strategy reaching 0.66 maximum robustness and 0.61 average robustness. Note that the 'top-down' strategy requires the search space adaptations to achieve this robustness value. Without adaptations, it only reaches the value of 0.5. 'Presbyopic' and 'top' follow with 0.54 and 0.49 average robustness respectively as shown in Figure 5b. Note that, although the 'top' strategy does not achieve high connectedness and connectivity, its robustness is high. The selection of highly ranked parents together with the effect of adaptations guarantee that the few established connections are highly robust.

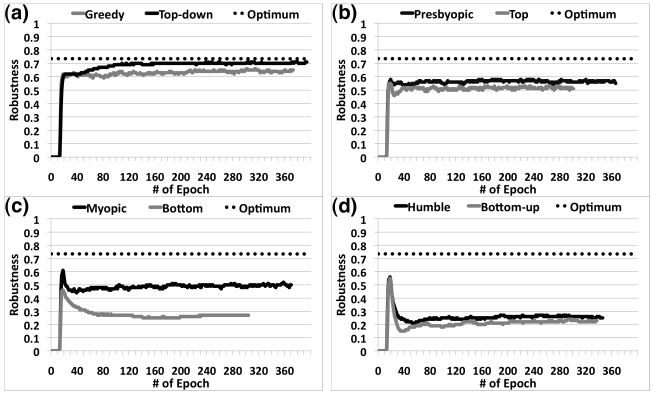


Figure 5. Convergence of robustness for the eight adaptation strategies. For these experimental settings, the optimum robustness is 0.74. (a) 'Top-down' and 'greedy'. (b) 'Top' and 'presbyopic'. (c) 'Myopic' and 'bottom'. (d) 'Humble' and 'bottom-up'.

The 'myopic' and 'bottom' strategies, illustrated in Figure 5c, do not provide highly robust topologies. The average robustness achieved is 0.47 and 0.27 respectively. Finally, 'humble' and 'bottom-up' perform the worst as shown in Figure 5d, with 0.25 and 0.21 average robustness respectively. Selecting the low ranked parents does not benefit the leaves of the tree. In this case, the agents that are in close proximity are most probably siblings (or belong to the same level in the tree), rather than potential parents in the level above.

The instability of the strategies in the formed topology is measured as well. Note that, for example, an average

instability value of $1/d = 1/5 = 0.2$ means that, on average, agents add or remove one connection in every epoch during runtime. The lowest instability of 0.1 is achieved by the 'bottom' strategy as it hardly introduces any connections. As mentioned in Section III, the system is 'trapped' in a local optimum of this strategy. Agents in the 'bottom' strategy try to connect with the lowest ranked children. This acts as an opposing force to self-organization goal.

The 'top-down' strategy achieves a very low instability of 0.12. Given that the 'top-down' strategy performs the best in the illustrated experiments, its low instability means that the system requires a minimal number of adaptations and changes in the connections to build the topology. The 'top', 'presbyopic', 'greedy' and 'myopic' strategies follow with 0.13, 0.13, 0.14 and 0.17 instability values respectively. Finally, 'humble' and 'bottom-up' have the highest instability of 0.2. Note that, the average instability of agents for all strategies is higher ($s \approx 0.2$) for agents with rank values around 0.7 and higher. These are the ones that should be connected with the leaves according to self-organization goal.

Finally, there is a significant difference in communication cost of the strategies. The 'bottom-up' and 'myopic' strategies have the highest communication cost with 7883 and 6703 average number of messages per epoch. This means that these two strategies heavily rely on adaptations for building the tree topology. Therefore, a high number of connections and disconnections are performed during runtime. The 'top', 'humble', 'top-down' and 'greedy' strategies follow with 5967, 5838, 5453 and 5446 average number of messages per epoch. The 'presbyopic' strategy consumes 4297 average number of messages per epoch. Adaptations are not effective in this case and therefore, agents do not perform easily connections and disconnections. The selected parents and children are ranked either close to 0 or 1, bounded to the endpoints of the range of ranks in the search space. The same holds for the 'bottom' strategy that generates on average 2359 messages per epoch.

Figure 6 visualizes the generated topologies for every strategy at four time points[4]. The topology generated by the 'top-down' strategy clearly resembles the optimum topology the most. The topology of the 'top-down' strategy is the most balanced and has the fewest disconnected agents. The 'greedy' strategy also generates a robust topology, however, a higher number of disconnected agents are present. The 'myopic', 'bottom-up' and 'humble' strategies are characterized by tree topologies with 'long' branches that correspond to groups of non-optimal connections. The 'presbyopic' and 'top' strategies depict a different topological feature. The forest is clustered in two large trees. Both seem balanced and robust to a high degree as Figure 5b also indicates. However, the two trees do not manage to be merged and, therefore, the connectedness and connectivity of these strategies are low. Finally, no tree can be distinguished in the forest of the 'bottom' strategy. The agents are either disconnected or form small branches.

[4]In Figure 6, the blue agents denote the ones that form the main tree. In contrast, the red agents represent the ones that are disconnected from the main body of the tree.

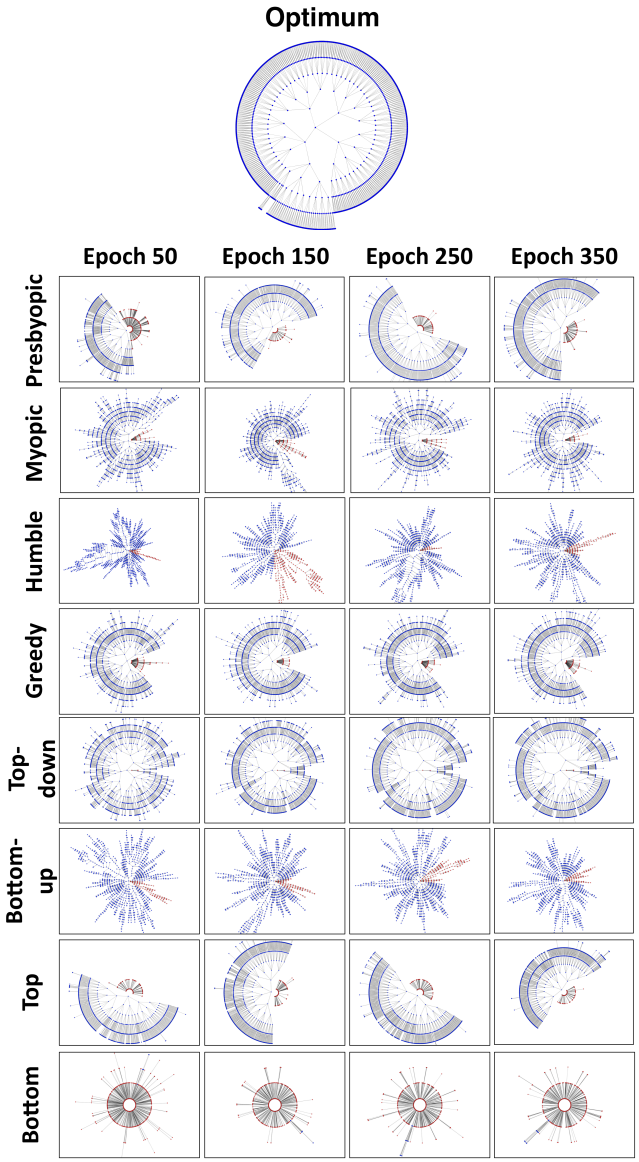


Figure 6. Visualization of the optimum tree topology and graphs of the eight adaptation strategies during convergence. Graphs are visualized in a radial layout that depicts the quality of the trees.

In all strategies, adaptation plays a crucial role, especially as far as robustness is concerned. Without adaptations, the 'top-down' strategy, which performs the best in the illustrated experiments, only reaches 0.5 maximum robustness.

VI. Discussion and Future Work

Results reveal that given the self-organization goal of Section III, the 'top-down' strategy performs the best for the four performance metrics this paper distinguishes. Communication cost is also low but not the minimum one compared to other strategies.

Two performance trade-offs are identified for the other strategies:

- **Communication cost versus connectivity**: 'Presbyopic' has a low communication cost but also low connectivity. 'Bottom-up' has high communication cost but also high connectivity.
- **Connectedness versus robustness**: 'Greedy' achieves low connectedness but high robustness. 'Myopic' achieves high connectedness but low robustness.

The 'bottom' strategy is inappropriate for this self-organization goal. However, note that, this strategy has a minimum communication cost and therefore diminishes the negative effects of its adoption.

A. Self-organization

Self-management of tree topologies based on the proposed adaptation strategies extends the concept of using a single and static fitness function to organize and optimize such topologies. Most existing self-organization mechanisms [7], [8], [9], [10] are based on a fitness function as part of their design. In these cases, agents are designed with one preference scheme on the basis of which neighbors are selected. This restricts their ability to generate tree topologies with different effects without either redesigning or introducing a new fitness function.

This paper explores eight adaptation strategies for tree topologies. Other topologies and self-organization goals are focus of current research. In most self-organization methodologies, preferences of agents and, therefore, their collective behavior, define the resulting topology. This means that by introducing and defining the concept of a measured ranking distance between agents in other topologies, adaptation strategies are potentially able to determine the preferences of agents and how the ranking distance is measured. This is topic of future research.

B. Adopting Adaptation Strategies

This paper shows the effects of adaptation strategies for a defined self-organization goal. An agent adopts an adaptation strategy as a system parameter within a homogeneous system of agents that all adopt the same strategy. This is the first step for showing the applicability of the adaptation strategies.

However, new possibilities arise when the adoption of a strategy is dynamic and can change during runtime. Which factor defines the adoption of a certain strategy? How can strategies be combined and work in synergy to achieve a new complex self-organization goal? Can a system of agents that initially begins with abstract strategy adoption collectively converge to a specific strategy or to a subset of the initial strategies? The answers to these questions in our future work can provide new possibilities for the applicability of adaptation strategies.

C. Applications

Different applications require different tree topologies to increase their performance. Adaptation strategies have different effects to the resulting trees. Therefore, instead of classifying the relevance of self-organization methods for trees to a range of distributed applications, classification can be abstracted according to the relevance of adaptation strategies.

The self-organization goal set in this paper is relevant to various applications, for example, application-level multicasting [1]. Case-studies of the use of adaptation strategies in such application domains can provide more insights about the influence of a strategy in the actual application performance.

VII. Conclusions

This paper proposes eight adaptation strategies that individual autonomous agents deploy to maintain a dynamic tree topology. A tree topology with different effects emerges for each adaptation strategy that agents deploy. These effects are evaluated based on four introduced metrics: connectedness, connectivity, instability and robustness. Simulation with 1500 agents shows that the 'top-down' strategy performs the best according to a defined self-organization goal and the four performance metrics. Comparisons of the results show specific performance trade-offs that can be made given the requirements of a domain of application.

The concept of adaptation strategies motivates us to further explore their applicability to other topologies, self-organization goals and methodologies. Dynamic adoption of strategies can provide new possibilities for self-organization. Studying these issues based on case studies in various application domains will provide new insights about the applicability of adaptation strategies in self-managed distributed networks.

Acknowledgment

This project is supported by the NLnet Foundation http://www.nlnet.nl.

References

[1] G. Tan, S. A. Jarvis, X. Chen, and D. P. Spooner, "Performance Analysis and Improvement of Overlay Construction for Peer-to-Peer Live Streaming," *Simulation*, vol. 82, no. 2, pp. 93–106, 2006.

[2] E. Pournaras, M. Warnier, and F. M. T. Brazier, "Local Agent-based Self-stabilisation in Global Resource Utilisation," *International Journal of Autonomic Computing (IJAC)*, 2010, (to appear).

[3] G. Strbac, "Demand side management: Benefits and challenges," *Energy Policy*, vol. 36, no. 12, pp. 4419–4426, December 2008.

[4] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "Baton: a balanced tree structure for peer-to-peer networks," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 661–672.

[5] E. Pournaras, M. Warnier, and F. M. T. Brazier, "Adaptive Agent-based Self-organization for Robust Hierarchical Topologies," in *ICAIS '09: Proceedings of the International Conference on Adaptive and Intelligent Systems*. IEEE, September 2009.

[6] E. Pournaras, M. Warnier, and F. M. Brazier, *Self-optimized Tree Overlays using Proximity-driven Self-organized Agents*, complex intelligent systems and their applications ed., ser. Optimization and its Applications. Springer, 2010, ch. 7, (to appear).

[7] M. Amoretti, "A framework for evolutionary peer-to-peer overlay schemes," in *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 61–70.

[8] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems," in *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 15.

[9] "T-man: Gossip-based fast overlay topology construction," *Computer Networks*, vol. 53, no. 13, pp. 2321 – 2339, 2009, gossiping in Distributed Systems.

[10] D. Miorandi, L. Yamamoto, and P. Dini, "Service evolution in bio-inspired communication systems," *ITSSA*, vol. 2, no. 1, pp. 51–60, 2006.

[11] D. Baldwin and G. Scragg, *Algorithms and Data Structures: The Science of Computing (Electrical and Computer Engineering Series)*. Rockland, MA, USA: Charles River Media, Inc., 2004.

[12] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*. London, UK: Springer-Verlag, 1998, pp. 601–610.

[13] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Trans. Comput. Syst.*, vol. 25, no. 3, p. 8, 2007.

[14] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "ProtoPeer: a P2P toolkit bridging the gap between simulation and live deployement," in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–9.

[15] J. Madadhain, D. Fisher, P. Smyth, S. White, and Y. Boey, "Analysis and visualization of network data using jung," *Journal of Statistical Software*, vol. 10, pp. 1–35, 2005.