Understanding the Information Flow of ACO-Accelerated Gossip Algorithms

Andreas Janecek(✉) and Wilfried N. Gansterer

University of Vienna, Vienna, Austria
{andreas.janecek,wilfried.gansterer}@univie.ac.at

Abstract. Gossip algorithms can be used for computing aggregation functions of local values across a distributed system without the need to synchronize participating nodes. Very recently, we have proposed acceleration strategies for gossip-based averaging algorithms based on ant colony optimization, which reduce the message and time complexity of standard gossip algorithms without additional communication cost. In this paper, we extend our latest studies by analyzing in detail how the proposed acceleration strategies influence the node selection of different variants of PushPull gossip algorithms and show that the directions of information dissemination across the network differ strongly according to the type of the underlying "knowledge" of the neighbors (local vs. global knowledge). This analysis leads to a better understanding of how information is spread throughout the network and provides important insights that can be used to further enhance the acceleration strategies.

Keywords: Applications of ant colony optimization · Gossip-based averaging · Epidemic aggregation · Acceleration

1 Introduction and Related Work

In distributed averaging tasks, the goal is to calculate the average $\bar{v}$ of a set of initial values v_i stored locally at the n nodes. Depending on the application, v_i could be sensor measurements, document attributes, media ratings, etc. One possibility for computing $\bar{v}$ in a completely distributed manner is to use gossip (or "*epidemic*") algorithms. Because of their potential robustness and inherently probabilistic nature (randomized communication schedules), these algorithms have the potential for tolerating dynamic network changes, node failures, or data loss and thus for providing a high degree of resilience and fault tolerance [1], allow for gradually trading runtime performance against communication cost, fault tolerance, energy consumption, and even privacy protection by adapting the intensity and regularity of interaction between nodes. Although several (theoretical) studies have proven that gossip averaging algorithms scale well with the number of nodes n, most of these studies are restricted to fully connected networks and based on rather strong assumptions. Applying gossip algorithms on non-fully connected networks significantly increases the number

Y. Tan et al. (Eds.): ICSI 2016, Part I, LNCS 9712, pp. 426–433, 2016.
DOI: 10.1007/978-3-319-41000-5_42

of messages / rounds, especially on weakly connected networks without a regular structure. Distributed aggregation methods can be accelerated, e.g., based on classical convergence acceleration techniques or Chebyhsev acceleration with time-dependent coefficients [2]), via exploiting additional assumptions on the distributed environment, e.g., based on the optimization of communication patterns with respect to a fixed topology [3] or topology itself [4], or by assuming that nodes have additional global knowledge about the topology and their position and by using multi-hop communication, substantial improvements can be achieved [5,11]. However, often a more general setup without such additional assumptions and only nearest-neighbor communication is required. Very recently [6], we have proposed different acceleration strategies based on ant colony optimization (ACO) in order to improve the diffusion speed of single-hop gossip averaging. The pheromone concept of ACO is adapted such that each node maintains a pheromone deposit for each outgoing link, which influences the probability of selecting a neighboring node as communication partner. Moreover, the (inverse) concept of pheromone evaporation is included to increase the attractiveness of nodes which have not been chosen for a long time. Just as in original gossip, our accelerated versions are based on single-hop communication, where every node only communicates with its direct neighbors without any overlay network. The application of SI in distributed environments is motivated by the fact that many natural examples of SI are based on (simple) individuals that communicate to develop collective behavior in a purely decentralized and self-organized fashion. These characteristics make these natural systems robust to loss of members and adaptable to a changing problem domain — all properties which are highly demanded in the context of distributed computing environments. Due to the underlying distributed setting, the pheromone update in [6] has to be performed locally on each node. The learning encoded in the pheromones is not directly based on the pseudo-random proportional ACO update rule but rather on the (dis-)similarity of the estimates of neighbors compared to the local estimate of nodes. Moreover, our "ants" are restricted to local movements only.
Contributions. We extend our study in [6] with an evaluation of the behavior of our acceleration strategies by analyzing how the proposed acceleration strategies influence the node selection of different variants of PushPull gossip algorithms. Additionally, we show that the directions of information dissemination across the network differ strongly according to the type of the underlying "knowledge" of the neighbors (local vs. global knowledge).

2 Gossip-Based PushPull Averaging

PushPull averaging is based on *exchange* of information (cf. [7]). In the active thread, each node i selects a random neighbor p as communication partner and pushes its current local estimate. Node p receives the packet, replies with its own current estimate (*pull-reply*), and stores the average of the received and its own estimate as its new estimate. Finally, the sender receives the answer and updates its own local estimate. There exists another group of gossip algorithms which are based solely on push averaging. The *PushSum* algorithm ([8]) needs more rounds than PushPull to converge but has the benefit that it preserves the mass conservation invariant — at any time the sum of all values (*i. e.,* approximations) in the network remains constant throughout the algorithm. This guarantees the correctness of algorithm even if messages are delayed [9] — a very important property for distributed systems. The mass conservation invariant can be violated in basic PushPull algorithms if an atomic violation happens, *i. e.,* if a node receives a push while it is waiting for a pull-reply. In order to deal with this problem, an enhanced version of PushPull called Symmetric PushSum Protocol (SPSP, [10]) can be used instead whenever mass conservation needs to be preserved. If there are no atomic violations, SPSP is identical to PushPull. Since PushPull is much simpler than SPSP, we focus on PushPull in the rest of this paper. However, whenever mass conservation needs to be preserved, SPSP can be used instead. PushPull/SPSP can be implemented as purely round-based or round *and* event-based implementation. Moreover, the *update process* may have a significant influence on the number of rounds and on the total number of messages necessary to achieve a given accuracy. If all nodes send their messages at the same time — as often stated in the literature ([7]) — the update of received information can only be performed after all nodes have finished sending in each round. If all nodes send their packets at slightly different times, it is possible that the update based on received information is performed *before* a node performs sending in the current round. A detailed evaluation of these different implementation strategies can be found in [6]. Here, we focus on the round-based PushPull strategy with immediate update, where each node actively sends one packet per round (and additionally replies with a pull-reply message if it receives a push message). Whenever a node p (the receiver) receives a (push) message from node i (the sender), the current local value of p is stored in a temporary variable tx_p and the local value of p is updated with received local value of i. Finally, the local value of i is updated with the temporarily stored previous local value of p. Recall that a delayed update will probably lead to an atomic violation.

3 Acceleration Based on ACO

Consider that each node in the network maintains a pheromone deposit for each outgoing link, as typical in ACO. The amount of pheromone along a directed link influences the probability of selecting a neighbor as communication partner. Our goal is to accelerate the diffusion speed of gossip algorithms by selecting links with higher pheromone value with higher probability. We describe how the amount of pheromone along a path is calculated, and how the (inverse) concept of evaporation is included. The variable λ_i refers to the local estimate at node i, i.e., $\lambda_i = x_i$. We discuss the acceleration strategies for PushPull, although the same ACO-based communication partner selection strategy can be applied for SPSP. At all times t, every node i has a current estimate λ_i of the average $\bar{v}$. Beyond that, node i also has (possibly outdated) information about the estimates of its neighbors, stored in the vector $\boldsymbol{y}_i$ of length $deg(i)$. The elements of this vector are ordered according to the IDs of the neighbors of i.

Example. Consider a node a connected to nodes b, c, d, e. Whenever node a communicates with one of its neighbors, it updates not only its own estimate λ_a, but also $\boldsymbol{y}_a$. The absolute difference between $\boldsymbol{y}_a$ and λ_a is denoted as $\boldsymbol{d}_a$ (*i. e.,* $\boldsymbol{d}_a = |\boldsymbol{y}_a - \lambda_a \mathbf{1}|$), and serves as the basis for our biased communication partner selection. Motivated by the concept of ACO, $\boldsymbol{d}_a$ represents the intensity of the pheromone trail along the edges between a and its neighbors. Node a will now choose edges with higher pheromone values with higher probability, i.e., it is more likely that node a selects a node with a strongly different local estimate than a node whose local estimate is very similar. The rationale behind is that more progress towards the true average will be made if an information exchange brings more new information. Clearly, since only ***local knowledge*** about neighbors is available, most elements in $\boldsymbol{y}_a$ and therefore also $\boldsymbol{d}_a$ will be outdated since a does not always know the true current estimate of all of its neighbors. E.g., consider that in round t nodes b and a exchange information, and that in round $t+1$ node a exchanges information only with node c while b exchanges information with another node. At the end of round $t+1$ the information of node a about the estimate of b is outdated and probably differs (at least slightly) from $b's$ current estimate. However, as we will see later, even partly outdated estimates are better than basic PushPull without any information about the neighbors. In ACO, the attractiveness of a pheromone trail is reduced as the pheromones evaporate over time. We exploit this strategy in the opposite direction and increase the attractiveness of edges over time in order to increase the attractiveness of nodes which have not been chosen for a long time. Whenever node a communicates with a neighbor — either as active sender *or* as receiver — it stores the number of the current round in the vector $\boldsymbol{t}_a$. The elements of $\boldsymbol{t}_a$ are also ordered according to the IDs of the neighbors of a. In our example, $\boldsymbol{t}_a(1)$ contains the information in which round the latest information exchange between a and b happened, independently of which of the two nodes initiated the information exchange. For all gossip algorithms, ACO-based acceleration can be implemented using a *roulette-wheel selection or a greedy selection strategy:* The vectors $\boldsymbol{d}_a$ and $\boldsymbol{t}_a$ are used to calculate $\boldsymbol{p}_a$ according to $\boldsymbol{p}_a = \boldsymbol{d}_a^{\alpha} \otimes (t\mathbf{1} - \boldsymbol{t}_a)$, where exponentiation is meant element-wise (the parameter α can be used emphasize edges with high pheromone trails), the symbol $\otimes$ represents element-wise multiplication, and t refers to the number of the current round. After normalization, $\boldsymbol{p}_a$ contains probabilities for selecting each neighboring node based on a *roulette-wheel* selection. Additionally, a *greedy* strategy can be used to select the node with the most different estimate. All values of $\boldsymbol{d}_a$ which are smaller than the maximum value of $\boldsymbol{d}_a$ are set to 0 before calculating the vector $\boldsymbol{p}_a$ in the above equation. If there is only one unique maximum value in $\boldsymbol{d}_a$, this node will be selected, otherwise the roulette wheel selection is used (*i. e.,* frequency of iteration is also an issue in this case).

Overhead. There is only a small overhead compared to basic gossip averaging since there is no additional communication. Only local computation and the following local memory space are required: at each node i, two additional vectors

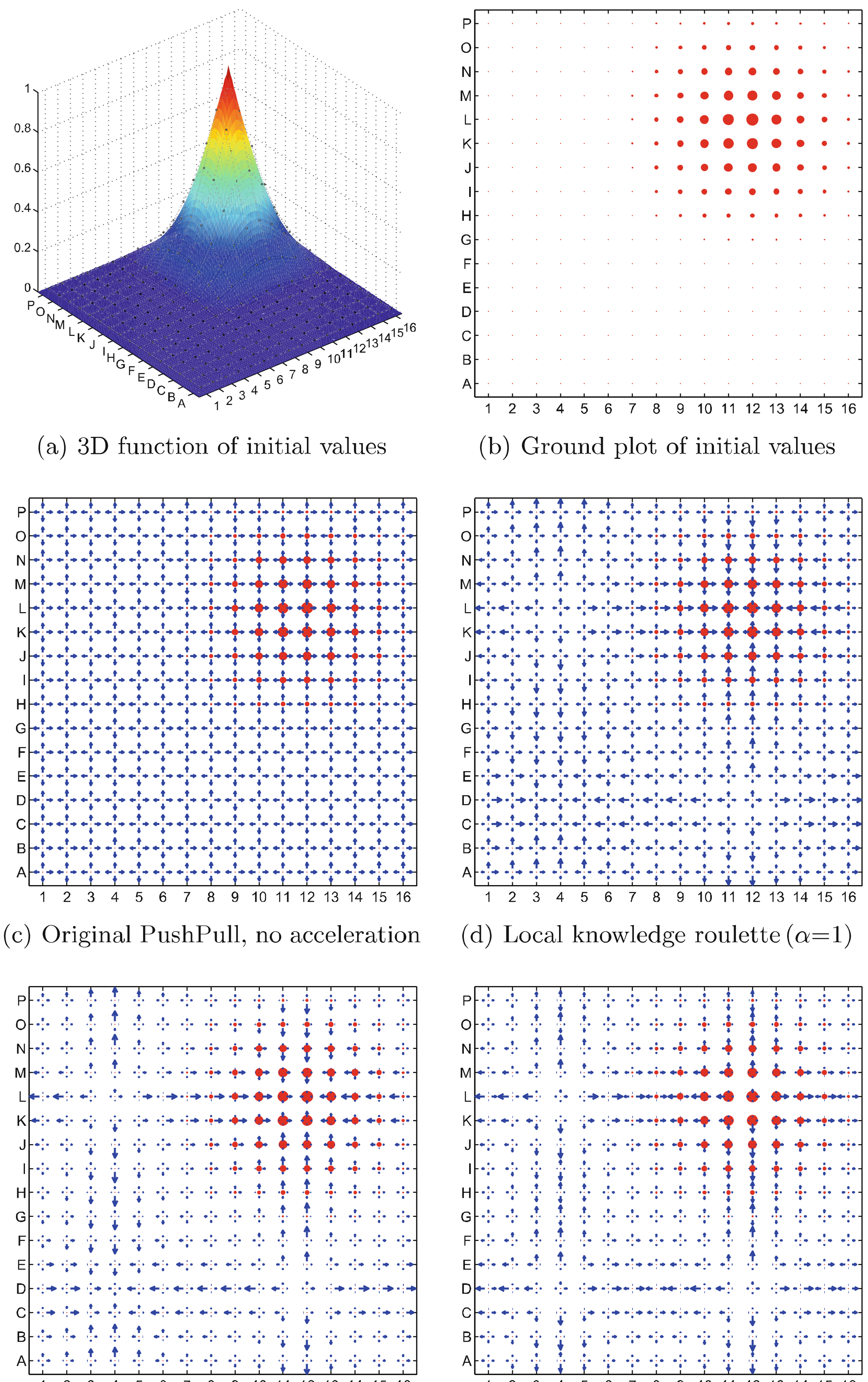
$\boldsymbol{y}_i$ and $\boldsymbol{t}_i$ with an average length of d_{avg} (the average node degree) must be stored.

ACO-based Acceleration Using Global Knowledge (reference algorithm): In order to demonstrate the best possible results that can be achieved with our acceleration strategies, we have simulated our algorithms based on the assumption that *perfect (global) knowledge* of the current estimates of all neighbors is available at all nodes. Technically, this can be simulated by replacing possibly outdated values in vector $\boldsymbol{y}_i$ with the current estimates of all neighboring nodes.

4 Evaluation and Analysis of Acceleration Strategies

All algorithms are implemented in Matlab in a simulation that allows for executing different algorithmic variants while being able to monitor the state of the network at any time from a bird's-eye-view perspective. The error of the estimated average is calculated after each round as $||\boldsymbol{v}(t) - \bar{v}\mathbf{1}||_2 \, / \, ||\boldsymbol{v}(0)||_2$, where, $\boldsymbol{v}(t)$ is the n-dimensional vector of all estimates after round t and $\boldsymbol{v}(0)$ is a vector consisting of the initial estimates (cf. [11]). All algorithms are terminated once the error is less than 10^{-8} (single precision). A detailed experimental evaluation of the acceleration strategies can be found in [6]. This evaluation includes (*i*) different network topologies / sizes, (*ii*) different initialization fields, (*iii*) four different gossip algorithms, (*iv*) purely round-based as well as round- and event-based implementations, and (*v*) acceleration strategies based on local and global knowledge with roulette-wheel and greedy selection strategies. Summarizing the results, we can say that not only the average node degree significantly influences the amount of acceleration, but also the irregularity of the topology. The acceleration strategies work best for weakly connected, irregular networks. In the optimal case an acceleration factor of up to 2.7 can be observed.

Analysis. In the following, we analyze in detail how the proposed acceleration strategies as well as the reference strategies influence the node selection of one representative gossip algorithm. Figure 1 shows the influence of the ACO-based acceleration on the selection of communication partners for a 2D-torus graph with 256 nodes. The number of rounds was set to 400 for all algorithms, and all algorithms were terminated when the error dropped below 10^{-8}. The results are average values over 10 runs. For larger networks, the results are similar but more difficult to visualize. Figure 1(a) and (b) show the initialization fields used for creating the plots in Fig. 1(c–f); large red dots in Fig. 1(b) refer to initialization values close to 1, and small red dots to initialization values close to 0. We used an initialization with a peak shifted towards the upper right corner in order to better visualize the important information. For being able to reference specific nodes, we labeled the grid with numbers from 1 to 16 along the x-axis, and with letters from A to P along the y-axis. The size of the blue arrows between any two nodes in Fig. 1(c–f) indicates how often nodes have communicated with each other. For example, the arrow pointing from node $A1$ to node $A2$ indicates how often node $A2$ has been selected as communication partner by node $A1$, and vice versa. Note



(a) 3D function of initial values

(b) Ground plot of initial values

(c) Original PushPull, no acceleration

(d) Local knowledge roulette (α=1)

(e) Local knowledge, greedy (max)

(f) Global knowledge, greedy (max)

Fig. 1. Influence of acceleration strategies on a 2D-torus with 16×16 nodes (Color figure online)

that there is also an arrow from node $A1$ to node $A16$ (the arrow left of node $A1$), and one arrow from node $A1$ to node $P1$ (the arrow underneath node $A1$). This is due to the fact that Fig. 1 shows the results for a 2D-torus graph. The size of the arrows is proportional to the communication frequency along this link — a large arrow indicates that a node has been selected frequently, while a small arrow indicates that a node has been chosen only rarely. Note that each node selects one communication partner per round, although Fig. 1(e) and (f) give the impression that some nodes select more communication partners than others. In fact, the size of the arrows is chosen such that higher-than-average values are represented with large arrows, while the arrow size for average and smaller-than-average values varies only slightly. Moreover, arrows are normalized w.r.t. the largest arrow for each figure. Figure 1(c) refers to the original PushPull algorithm without acceleration. As expected, the selection of nodes is uniformly distributed since each node is connected to four neighboring nodes and all nodes are always selected with the same probability. Figure 1(d–f) illustrate how this distribution changes when the ACO-based acceleration strategy is applied. The results using *local* knowledge are shown in Fig. 1(d) for the roulette-wheel selection strategy with $\alpha = 1$, and in Fig. 1(e) for the greedy selection strategy. The results for the reference algorithm with *global* knowledge using the greedy selection strategy are shown in Fig. 1(f). Since the results in Fig. 1(d) are a hybrid between Fig. 1(c) and (e), we focus on Fig. 1(e) and (f). It is interesting to observe the patterns in Fig. 1(e) and (f): in both cases, there are patterns similar to a "number sign" ("#"), whose lanes intersect at $D4$, $L4$, $D12$, and $L12$. The intersection point at $L12$ is located at the peak of the initialization field; all other intersection points are located at nodes with maximal distance to $L12$: $L4$ and $D12$ are located eight single-hops away from the node at $L12$, and $D4$ is located eight single-hops away from $L4$ and $D12$, respectively. The reason for this special shape is the regular diffusion of information in a rectangular 2D-torus graph (such regular behavior can also be observed for hypercubes, however, this information cannot be displayed in a 2D plot). We point out that the "directions" of these lanes differ between Fig. 1(e) and (f). In Fig. 1(e) (local knowledge), there are two intersections at $L4$ and $D12$ with arrows pointing *away* from them, and two intersections at $D4$ and $L12$ where arrows are pointing *towards* them. In Fig. 1(f) (global knowledge), the arrows tend to point away from *all* intersection points. Indeed, the plot in Fig. 1(f) could be decomposed into four identical squares, which is not the case for Fig. 1(e). One explanation for this difference is the fact that in Fig. 1(e) the neighboring nodes of $D4$ and $L12$ tend to have outdated information about the estimates of $D4$ and $L12$ and chose these nodes more often than others. However, the diffusion of information for these two acceleration strategies is very similar and there are also only small variations in the speed and direction of information diffusion in the network. This analysis provides interesting insights how information diffuses throughout the network. In our current research we aim at applying the observed distribution of node selection from our acceleration algorithms based on global knowledge (Fig. 1(f)) on gossip algorithms with only local knowledge. The goal is to further increase the diffusion speed – however, without the need for exact knowledge of the neighbors' estimates.

5 Conclusions

We have extended our study on ACO-based acceleration strategies for gossip-based averaging algorithms, and evaluated the behavior of our acceleration strategies by analyzing how they influence the node selection of different variants of PushPull gossip algorithms. We have shown that the directions of information dissemination across the network differ strongly according to the type of the underlying "knowledge" of the neighbors (local vs. global knowledge). This analysis leads to a better understanding of how information is spread throughout the network and provides important insights that can be used to further enhance the acceleration strategies. *This research has been partially supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-113.*

References

1. Montresor, A.: Designing extreme distributed systems: challenges and opportunities. In: Proceedings of the 8th ACM SIGSOFT Conference, pp. 1–2. ACM (2012)
2. Golub, G.H., Varga, R.S.: Chebyshev semi-iterative methods, successive overrelaxation iterative methods. Numer. Math. **3**, 157–168 (1961)
3. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE T. Inform. Theory **52**, 2508–2530 (2006)
4. Kar, S., Moura, J.M.F.: Sensor networks with random links: topology design for distributed consensus. IEEE T. Signal Proces. **56**, 3315–3326 (2008)
5. Benezit, F., Dimakis, A., Thiran, P., Vetterli, M.: Order-optimal consensus through randomized path averaging. IEEE Trans. Inform. Theory **56**, 5150–5167 (2010)
6. Janecek, A., Gansterer, W.N.: Aco-based acceleration of gossip averaging. In: GECCO (2016). http://dx.doi.org/10.1145/2908812.2908832
7. Jelasity, M.: Gossip. Self-organising Software. Springer, Heidelberg (2011)
8. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Symposium on Foundations of Computer Science, pp. 482–491. IEEE (2003)
9. Jesus, P., Baquero, C., Almeida, P.S.: Dependability in aggregation by averaging. CoRR abs/1011.6596 (2010)
10. Blasa, F., Cafiero, S., Fortino, G., Di Fatta, G.: Symmetric push-sum protocol for decentralised aggregation. In: AP2PS, IARIA, pp. 27–32 (2011)
11. Dimakis, A., Sarwate, A., Wainwright, M.: Geographic gossip: Efficient averaging for sensor networks. IEEE T. Signal Proces. **56**, 1205–1216 (2008)