Brief Announcement: Towards Round-Optimal Approximate Agreement on Trees

Marc Fuchs
University of Freiburg
Freiburg, Germany
marc.fuchs@cs.uni-freiburg.de

Diana Ghinea*
Lucerne University of Applied Sciences and Arts
Zug, Switzerland
diana.ghinea@hslu.ch

Zahra Parsaeian
University of Freiburg
Freiburg, Germany
zahra.parsaeian@cs.uni-freiburg.de

Abstract

Approximate Agreement (AA) is a key consensus primitive that allows honest parties to achieve close but not necessarily identical outputs, even in the presence of Byzantine faults. While optimal round complexity for synchronous AA on real values is well understood, its extension to other input spaces remains an open problem.

We present a protocol achieving AA on trees in the synchronous model, with round complexity $O\left(\frac{\log |\mathsf{V}(T)|}{\log\log |\mathsf{V}(T)|}\right)$, where $\mathsf{V}(T)$ is the set of vertices in the input space tree T. Our protocol non-trivially reduces the problem of AA on trees to AA on real values.

Additionally, we extend the impossibility result regarding the round complexity of AA protocols on real values to trees: we prove a lower bound of $\Omega\left(\frac{\log \mathsf{D}(T)}{\log\log \mathsf{D}(T)}\right)$ rounds, where $\mathsf{D}(T)$ denotes the diameter of the input space tree. This establishes the asymptotic optimality of our protocol for trees of large diameter $\mathsf{D}(T) \in \Theta(\left|\mathsf{V}(T)\right|)$.

CCS Concepts

• **Theory of computation → Cryptographic protocols**; *Distributed algorithms.*

Keywords

approximate agreement, trees, optimal round complexity

ACM Reference Format:
Marc Fuchs, Diana Ghinea, and Zahra Parsaeian. 2025. Brief Announcement: Towards Round-Optimal Approximate Agreement on Trees. In *ACM Symposium on Principles of Distributed Computing (PODC '25), June 16–20, 2025, Huatulco, Mexico.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3732772.3733555

Related Version: A full version of this paper is available at [10].

1 Introduction

Ensuring consistency among parties in a distributed system is essential, yet it is a difficult task in the face of potential failures or malicious behavior. Agreement protocols serve as indispensable tools for achieving consensus in such environments. One such fundamental primitive is Approximate Agreement (AA).

The AA problem, as defined by [4], considers a setting of n parties $p_1, p_2, \ldots, p_n$ in a fully connected network where each party holds a real value as input. Even if up to t of the n parties are Byzantine (i.e., malicious), AA enables the honest parties to obtain values satisfying two conditions. First, **Validity**: the outputs of the honest parties must lie within the honest inputs' range. Second, for any predefined error $\varepsilon > 0$, **ε-Agreement**: the honest parties' outputs must be pairwise ε-close. This relaxed form of agreement has proven effective in scenarios where exact consensus is either unnecessary or infeasible, such as clock synchronization [13], blockchain oracles [1], distributed machine learning [6, 7, 20], aviation control systems [14, 19], or robot gathering [18] on various map structures.

The AA problem is not limited to real-valued inputs. Several variants have been explored, including multidimensional real inputs [15, 16, 21] and discrete domains such as lattices and various classes of graphs [17]. In this work, we focus on AA on trees. In this variant, as presented by Nowak and Rybicki [17], the input space is a labeled tree T that is known to all parties. Then, each party has a vertex of T as input, and every honest party must output a vertex of T. The Validity condition generalizes the requirement that honest outputs lie within the honest inputs' range to requiring that they lie within the honest inputs' *convex hull*—i.e., the smallest subtree of T that contains all honest input vertices. In addition, the ε-Agreement requirement becomes simply 1-Agreement: the distance between any two honest outputs must be at most one.

Regardless of the space considered, the AA problem admits very elegant solutions that follow a common iteration-based outline. In each iteration, the parties use some mechanism to distribute their current values (in the first iteration, these are the inputs) to all parties within $O(1)$ rounds of communication. Afterward, each party computes a new value such that: the new values are in the convex hull of the values distributed by the honest parties, and the new values get *closer*. This way, sufficient iterations, AA is achieved. Although this outline is a valuable tool for proving that AA can be solved for various input spaces and in various communication models, it leaves open questions regarding efficiency.

In terms of round complexity for real values, this outline incurs $O(\log(\mathsf{D}/\varepsilon))$ communication rounds if the honest inputs are D-close [4, 17]. However, Fekete [9] has shown that this is suboptimal for protocols designed in the synchronous model (where all messages are delivered within a publicly known amount of time, and parties have perfectly synchronized clocks), presenting a lower bound and an asymptotically matching protocol tolerating $t < n/4$ corruptions. Fekete's lower bound was later asymptotically

*This work was partially carried out while the author was at ETH Zürich.

PODC '25, Huatulco, Mexico

ACM ISBN 979-8-4007-1885-4/25/06
https://doi.org/10.1145/3732772.3733555

matched also for the optimal resilience thresholds $t < n/3$ [2] and, with cryptographic setup, $t < n/2$ [12]. These protocols deviate from the standard outline by utilizing information from previous iterations (i.e., identifying corrupted parties that attempt to send different values to different parties). Hence, optimal round complexity for AA on real values is well-understood, and the results naturally extend to simple input spaces such as paths. However, generalizations to more complex spaces remain an open problem. In this work, we build on the results for real values and extend them to trees. Specifically, we address the following question:

What is the optimal round complexity for solving AA on trees in the synchronous model?

Regarding prior solutions for AA on trees, the protocol of [17] follows the standard iteration-based outline and achieves round complexity $O(\log \mathsf{D}(T))$, where $\mathsf{D}(T)$ denotes the diameter of the input space tree T. We note that this protocol is designed in the asynchronous model (where messages get delivered *eventually* as opposed to within a publicly known amount of time). While subsequent works improved message complexity [8], the round complexity of $O(\log \mathsf{D}(T))$ remains the state-of-art in the asynchronous model, and in the synchronous model as well up to our work.

Our Contribution. We provide a synchronous protocol achieving AA on trees with round complexity $O\left(\frac{\log |\mathsf{V}(T)|}{\log\log |\mathsf{V}(T)|}\right)$, where $\mathsf{V}(T)$ denotes the set of vertices of the input space tree T. Our protocol relies on novel techniques that reduce the problem to AA on real values, and uses the protocol of [2] as a building block.

In addition, we explain how Fekete's bound can be adapted to trees: we show that, under the assumption that a constant fraction of the parties is corrupted, $\Omega\left(\frac{\log \mathsf{D}(T)}{\log\log \mathsf{D}(T)}\right)$ rounds are necessary for any AA protocol on tree T. Hence, our protocol achieves optimal round complexity for trees T of large diameter $\mathsf{D}(T) \in \Theta(|\mathsf{V}(T)|)$.

Open Problems. Our findings highlight promising directions for further work. First, our protocol achieves optimal round complexity for large-diameter trees, up to constant factors. Improving the constants in the round complexity of AA for real values would directly refine our protocol's efficiency, making it more practical for real-world applications. Second, it would be valuable to determine whether our lower bound on round complexity can be matched for trees T with low diameter $\mathsf{D}(T) \in o(|\mathsf{V}(T)|)$. Finally, it remains an open question whether similar round-optimal guarantees can be achieved for synchronous AA on broader classes of graphs.

2 Lower Bound

We recall Fekete's lower bound for AA on real values regarding *how close* the honest values may get after R rounds.

Theorem 1 (Theorem 15 of [9]). *Let Π denote an arbitrary deterministic R-round protocol achieving Validity and Termination on $\mathbb{R}$ up to t Byzantine corruptions. Then, given $a, b \in \mathbb{R}$ with $b - a \geq D$, there is an execution of Π where the honest inputs are in $\{a, b\}$ and two honest parties output values v and v' satisfying $|v - v'| \geq D \cdot \frac{t^R}{R^R \cdot (n+t)^R}$.*

In the full version of our paper, we explain how the proof technique of [9] can be adapted to the setting of trees and paths, leading to the following result:

Theorem 2. *Every deterministic protocol achieving AA on tree T even up to $t = \theta(n)$ Byzantine corruptions has round complexity $\Omega\left(\frac{\log D(T)}{\log\log D(T)}\right)$.*

3 Our Protocol

In the following, we present the high-level ideas and intuition behind our protocol TreeAA, described by the theorem below. For a complete presentation, see the full version of our paper.

Theorem 3. *There is a protocol TreeAA achieving AA even up to $t < n/3$ Byzantine corruptions on any input space tree T with round complexity $O\left(\frac{\log|V(T)|}{\log\log|V(T)|}\right)$.*

3.1 Warm-up: Protocol for Paths

Building towards our protocol TreeAA, we first describe a protocol for paths: the input space tree is a path P. We make use of the protocol of [2], denoted by RealAA, which achieves AA on $\mathbb{R}$ with asymptotically optimal round complexity. We add that the analysis presented in [2] assumes $\varepsilon = 1/n$. In the full version of the paper, we extend their analysis to any ε, obtaining the theorem below.

Theorem 4. *There is a protocol RealAA(ε) achieving AA on real values for $t < n/3$. If the honest inputs are D-close, RealAA(ε) ensures Termination within $R_{RealAA}(D, \varepsilon) = \left\lceil 7 \cdot \frac{\log_2(D/\varepsilon)}{\log_2\log_2(D/\varepsilon)} \right\rceil$ rounds.*

Protocol RealAA can be adapted to paths in a straightforward manner. The parties denote the k vertices in the input space path P by $(v_1, v_2, \ldots, v_k)$ (so that v_i, v_{i+1} are adjacent, and v_1 is the endpoint of P with the lowest label in lexicographic order). If a party's input v_{IN} is the vertex now denoted by v_i, it joins RealAA(1) with input i. Each party obtains a value $j \in \mathbb{R}$ from RealAA and may output the vertex denoted by $v_{\mathsf{int}(j)}$, where $\mathsf{int}(j)$ denotes the closest integer to j. That is, if $z \leq j < z+1$ for $z \in \mathbb{Z}$, $\mathsf{int}(j) := z$ if $j - z < (z+1) - j$ and $\mathsf{int}(j) := z + 1$ otherwise. We add the following remarks:

- If $i_{\min}, i_{\max} \in \mathbb{Z}$ and $j \in [i_{\min}, i_{\max}]$, $\mathsf{int}(j) \in [i_{\min}, i_{\max}]$.
- If $j, j' \in \mathbb{R}$ satisfy $|j - j'| \leq 1$, then $|\mathsf{int}(j) - \mathsf{int}(j')| \leq 1$.

As any two honest parties obtain 1-close values j, j', we obtain that $|\mathsf{int}(j) - \mathsf{int}(j')| \leq 1$. This implies that the vertices $v_{\mathsf{int}(j)}$ and $v_{\mathsf{int}(j')}$ have distance at most 1, and 1-Agreement holds. Therefore, we have achieved AA on P in $O\left(\frac{\log \mathsf{D}(P)}{\log\log \mathsf{D}(P)}\right)$ rounds.

3.2 Moving Towards Trees

We reduce the problem of solving AA on trees to AA on $\mathbb{R}$ as well. This section presents a stepping stone towards our final solution: as shown in Figure 1, we assume that the parties *know* a path P in the tree that intersects the honest inputs' convex hull. Then, the parties may proceed as follows: each party with input $v_{\text{IN}} \in \mathsf{V}(T)$ computes the projection of v_{IN} onto P, denoted by $\mathsf{proj}_P(v_{\text{IN}})$. This is the vertex in P that has the shortest distance to v_{IN}. Note that $\mathsf{proj}_P(v_{\text{IN}})$ is in the honest inputs' convex hull.

From this point, we may achieve AA using the approach described in Section 3.1. The parties denote the k vertices in P by $(v_1, v_2, \ldots, v_k)$, where v_1 is the endpoint with the lower label lexicographically. Each party joins RealAA with input i, where the vertex denoted by v_i in P is the projection $\mathsf{proj}_P(v_{\text{IN}})$ of the party's input

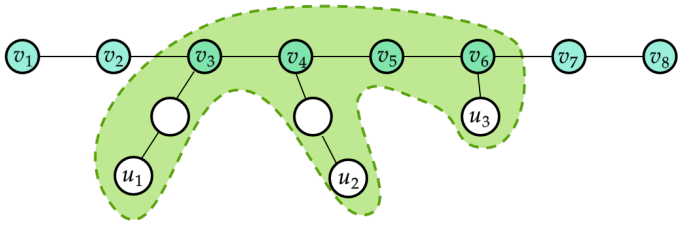

Figure 1: Let P be the assumed path, represented by the sequence of vertices $v_1, v_2, \ldots, v_8$. The vertices u_1, u_2, u_3 are the honest inputs, whose convex hull is highlighted in green. The projections of u_1, u_2, u_3 onto path P are vertices v_3, v_4, v_6 respectively.

vertex v_{IN}. Each party obtains an output $j \in \mathbb{R}$ and may output the vertex denoted by $v_{\text{int}(j)}$ in path P. These vertices are 1-close, and they are in the honest projections' convex hull, and therefore in the honest inputs' convex hull. Consequently, AA is achieved.

3.3 Finding a Path

Intuitively, it may seem that finding a path P that intersects the honest inputs' convex hull comes down to solving *Byzantine Agreement*. This would require $t+1 = O(n)$ rounds of communication [5], which generally prevents us from achieving our round-complexity goal. Instead, we implement a subprotocol PathsFinder that enables the honest parties to *approximately* agree on such a path. Concretely, each honest party will obtain a subpath P of T such that (i) P intersects the honest inputs' convex hull, and (ii) if two honest parties obtain two different paths P and P', then either P is P' with one additional edge, or P' is P with one additional edge, as described in Figure 2. Formally, if $P = (v_1, \ldots, v_k)$ and $P' = (u_1, u_2, \ldots, u_{k'})$, then either $P = P'$, or $P' = (v_1, \ldots, v_k, u_{k'})$, or $P = (u_1, \ldots, u_{k'}, v_k)$. This will suffice to use the approach of Section 3.2.

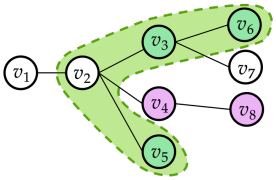

Figure 2: This figure shows a possible input space tree. Vertices v_3, v_5, v_6 are the honest inputs, and the green area highlights the honest inputs' convex hull. For this tree, PathsFinder may allow the honest parties to obtain as output, for instance, paths P such that either $P = (v_1, v_2, v_4)$ or $P = (v_1, v_2, v_4, v_8)$.

PathsFinder first defines a root vertex v_{root} for the input space tree T: this is the vertex with the lowest label in lexicographic order. Then, it enables the honest parties to obtain 1-Agreement on vertices in a subtree rooted at a vertex in the honest inputs' convex hull. We achieve this using a technique known for efficiently finding *lowest common ancestors* [3]. Each party runs a depth-first-search starting from the fixed root vertex v_{root}, and writes down each vertex whenever visited in the depth-first-search. In the example displayed in Figure 2, $v_{\text{root}} = v_1$, and the resulting list is $L = [v_1, v_2, v_3, v_6, v_3, v_7, v_3, v_2, v_4, v_8, v_4, v_2, v_5, v_2, v_1]$.

Note that, if i and i' are indices of list L that represent two honest inputs v and v', then vertices occurring in the list at indices $j \in [i, i']$ are in the subtree rooted at the lowest common ancestor of v and v': this is a vertex in the honest inputs' convex hull. Hence, if the honest parties run RealAA(1) using as input indices i representing their input vertices in list L, they obtain real values j such that: (i) the indices $\text{int}(j)$ in list L correspond to vertices in the subtree rooted at a vertex in the honest inputs' convex hull, and (ii) the indices $\text{int}(j)$ are 1-close and therefore they represent 1-close vertices. Then, each honest party may output the path connecting v_{root} to the vertex obtained through this process, which will ensure our desired guarantees. We add that each honest party obtains its path within $R_{\text{RealAA}}(2 \cdot |V(T)|, 1)$ rounds since the list obtained contains at most $2 \cdot |V(T)|$ vertices.

3.4 Putting It All Together

We may now describe our final protocol TreeAA. The parties first run PathsFinder to *approximately* agree on paths that intersect the honest inputs' convex hull. Afterward, each party p proceeds as described in Section 3.2: it denotes the k vertices on its own path P by $(v_1 := v_{\text{root}}, v_2, \ldots, v_k := v)$. It joins RealAA(1) with input i, where the vertex denoted by v_i in p's path P is the projection of p's input vertex v_{IN} onto P: $v_i = \text{proj}_P(v_{\text{IN}})$. Then, upon obtaining output j, p *should* output the vertex denoted by $v_{\text{int}(j)}$ in its path P. This is where we need to be careful about honest parties holding different paths P. If an honest party p obtains $j > k$, then p holds the "shorter" path $(v_1, \ldots, v_k)$, while other honest parties hold the "longer" path $(v_1, \ldots, v_k, v_{k+1})$. In this case, if $\text{int}(j) = k + 1$ and vertex v_k has at least three neighbors, p is unable to decide which neighbor is vertex v_{k+1}, i.e., the last vertex of the "longer path". Instead, in this case, p may simply output v_k.

A note on the $t < n/2$ case. We add that the resilience threshold of TreeAA is given by the underlying RealAA protocol assumed: whenever RealAA achieves AA on $[1, 2 \cdot |V(T)|]$, our protocol TreeAA achieves AA on the input space tree T, and the round complexity of TreeAA is double that of RealAA. Hence, our reduction also enables asymptotically optimal round complexity for trees of diameter $D(T) \in \Theta(|V(T)|)$ in *authenticated settings* (assuming digital signatures) up to $t = (1 - c)/2 \cdot n$ corruptions for any constant $c > 0$: we may simply replace RealAA with the Proxcensus protocol of [11].

References

[1] A. Bandarupalli, A. Bhat, S. Bagchi, A. Kate, C.-D. Liu-Zhang, and M. K. Reiter. 2024. Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles. In *Proceedings of the 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, Brisbane, Australia, 456–469. https://doi.org/10.1109/DSN58291.2024.00051

[2] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. 2010. Brief Announcement: Simple Gradecast Based Algorithms. In *Distributed Computing*, Nancy A. Lynch and Alexander A. Shvartsman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 194–197.

[3] Michael A. Bender and Martín Farach-Colton. 2000. The LCA Problem Revisited. In *LATIN 2000: Theoretical Informatics*, Gaston H. Gonnet and Alfredo Viola (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 88–94.

[4] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. 1986. Reaching Approximate Agreement in the Presence of Faults. *J. ACM* 33, 3 (May 1986), 499–516. https://doi.org/10.1145/5925.5931

[5] Danny Dolev and H. Raymond Strong. 1983. Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* 12, 4 (1983), 656–666.

[6] El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. 2021. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1918, 14 pages.

[7] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. 2020. Genuinely Distributed Byzantine Machine Learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 355–364. https://doi.org/10.1145/3382734.3405695

[8] Mose Mizrahi Erbes and Roger Wattenhofer. 2024. Asynchronous Approximate Agreement with Quadratic Communication. arXiv:2408.05495 [cs.DC] https://arxiv.org/abs/2408.05495

[9] Alan David Fekete. 1990. Asymptotically optimal algorithms for approximate agreement. *Distributed Computing* 4, 1 (1990), 9–29.

[10] Marc Fuchs, Diana Ghinea, and Zahra Parsaeian. 2025. Towards Round-Optimal Approximate Agreement on Trees. arXiv:2502.05591 [cs.DC] https://arxiv.org/abs/2502.05591

[11] Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. 2022. Round-Optimal Byzantine Agreement. In *Advances in Cryptology – EUROCRYPT 2022*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer International Publishing, Cham, 96–119.

[12] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2022. Optimal Synchronous Approximate Agreement with Asynchronous Fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 70–80. https://doi.org/10.1145/3519270.3538442

[13] Christoph Lenzen and Julian Loss. 2022. Optimal Clock Synchronization with Signatures. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 440–449. https://doi.org/10.1145/3519270.3538444

[14] Darya Melnyk and Roger Wattenhofer. 2018. Byzantine Agreement with Interval Validity. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE Computer Society, Salvador, Brazil, 251–260. https://doi.org/10.1109/SRDS.2018.00036

[15] Hammurabi Mendes and Maurice Herlihy. 2013. Multidimensional approximate agreement in Byzantine asynchronous systems. In *45th ACM STOC*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, Palo Alto, CA, USA, 391–400. https://doi.org/10.1145/2488608.2488657

[16] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K Garg. 2015. Multidimensional agreement in Byzantine systems. *Distributed Computing* 28, 6 (2015), 423–441.

[17] Thomas Nowak and Joel Rybicki. 2019. Byzantine Approximate Agreement on Graphs. In *33rd International Symposium on Distributed Computing (DISC 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 146)*, Jukka Suomela (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 29:1–29:17. https://doi.org/10.4230/LIPIcs.DISC.2019.29

[18] Maria Potop-Butucaru, Michel Raynal, and Sebastien Tixeuil. 2011. Distributed Computing with Mobile Robots: An Introductory Survey. In *Proceedings of the 2011 14th International Conference on Network-Based Information Systems (NBIS '11)*. IEEE Computer Society, USA, 318–324. https://doi.org/10.1109/NBiS.2011.55

[19] David Stolz and Roger Wattenhofer. 2016. Byzantine Agreement with Median Validity. In *19th International Conference on Principles of Distributed Systems (OPODIS 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 46)*, Emmanuelle Anceaume, Christian Cachin, and Maria Potop-Butucaru (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 1–14. https://doi.org/10.4230/LIPIcs.OPODIS.2015.22

[20] Lili Su and Nitin H. Vaidya. 2016. Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing* (Chicago, Illinois, USA) *(PODC '16)*. Association for Computing Machinery, New York, NY, USA, 425–434. https://doi.org/10.1145/2933057.2933105

[21] Nitin H. Vaidya and Vijay K. Garg. 2013. Byzantine vector consensus in complete graphs. In *32nd ACM PODC*, Panagiota Fatourou and Gadi Taubenfeld (Eds.). ACM, New York, NY, USA, 65–73. https://doi.org/10.1145/2484239.2484256