Discordant Voting Protocols for Cyclically Linked Agents

András Pongrácz

***Abstract*—Voting protocols, such as the push and the pull protocol, are designed to model the behavior of people during an election, but they have other applications. These processes have been studied in many areas, including but not limited to social models of interaction, distributed computing in peer-to-peer networks, and to describe how viruses or rumors spread in a community. In this paper we study the runtime of discordant linear protocols on the cycle graph, and the probability for each consensus to win in the end.**

***Index Terms*—Markov chain, voting protocol, cycle.**

I. Introduction

Models of voting in finite graphs have been studied intensively for decades, see e.g., [1], [2], [3], [4], [5], [6]. Throughout this paper, a discrete time voting protocol is defined by specifying a graph and a set of nondeterministic rules. Then the process is divided into rounds. In each round, the participants, i.e., vertices of the graph, can affect the vote of their neighbors according to the given rules.

We note that many alternative definitions were investigated in the literature. Continuous time voting processes were studied in [1], [7]. Somewhat surprisingly, the thorough mathematical investigation of the continuous version preceded that of the discrete analogue of the protocols [2], [7]. In [8] the graph evolves together with the opinions of the vertices. This models the behavior of people who in each round try to convince one another and succeed with a given probability. Whenever they fail, they cease to communicate with each other, that is, we delete the edge linking them from the graph. In such a model there are many potential final results, as the graph can disconnect, and in fact we may end up with many connected components. For more details, see [6], [9]. The application of these randomized protocols in studying how rumor spreads in a society goes back to decades, and it is still an active area [10], [3], [4]. The same can be said about peer-to-peer networks, see e.g., [11], [5], [12]. In this application, opinion is replaced by a piece of information that each computer has at a given time, and they share the data in a randomized way. Connections of voting processes and coalescing random walks were investigated in [7], [13], and for other recent applications see [14], [15].

However, we consider discrete time voting models where the graph is fixed, and the vote is a binary decision. The two options to choose from are 0 and 1, but we usually refer to vertices with opinion 0 as *blue* vertices, and *red* vertices are the ones with opinion 1. Such a protocol can be synchronous (see [16] for examples), i.e., it is allowed that several vertices of the graph change their opinion in one round; otherwise it is asynchronous. The so-called linear voting model was introduced in [16] as a common generalization of many well-studied voting protocols. Three of the most common special cases of asynchronous linear voting are the

- Oblivious protocol: in each round an edge uv is chosen uniformly at random, and then either u adopts the opinion of v or the other way around, with equal probability.
- Push protocol: in each round a vertex u is chosen uniformly at random, and that vertex forces a randomly chosen neighbor to adopt the opinion of u.
- Pull protocol: in each round a vertex u is chosen uniformly at random, and that vertex is forced by a randomly chosen neighbor v to adopt the opinion of v.

From a practical viewpoint, all linear voting models have a common weakness: it is typical that nothing changes in many steps of the process, as it is possible that every participant keeps his own opinion for the next round. E.g., consider push, pull or oblivious voting on the complete graph K_n; in this particular case, the three protocols coincide. If one opinion is significantly more popular than the other, then with very high probability, both chosen vertices have the more popular opinion. So usually many idle rounds go by before the opinion of some vertex is altered. This example demonstrates the advantage of discordant (oblivious, push, pull) voting protocols, defined in [17]. An edge uv is discordant if u and v have different opinion, and a vertex is discordant if it is in a discordant edge. To define discordant oblivious, push and pull voting, the above three definitions are modified so that whenever a random choice is made, we only allow discordant edges or vertices to be picked (always uniformly at random). Note that in our restricted framework when there are only two opinions, the definition of discordant pull voting simplifies to picking a discordant vertex in each round randomly and switching its opinion.

The goal of every voting scheme that we study now is to reach consensus, that is, a state where all participants have the same opinion. The topic of the present paper is the expected time T to reach consensus with the discordant push, pull and oblivious processes on the n-cycle. It was proven in [17] that all three processes have a quadratic runtime at worst. In particular, push voting is expected to terminate in at most $33n^2$ steps regardless of the initial state, and from some initial state it is indeed expected to take at least $n^2/4 + O(n)$ time to reach a unanimous vote [17, Section 4]. We improve the bounds and obtain the precise

Manuscript received March 6, 2018; revised March 28, 2018. This work is supported by the EFOP-3.6.2-16-2017-00015 project, which has been supported by the European Union, co-financed by the European Social Fund. The paper was also supported by the National Research, Development and Innovation Fund of Hungary, financed under the FK 124814 and PD 125160 funding schemes, and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

A. Pongrácz is with the Department of Algebra and Number Theory, University of Debrecen, Debrecen, 4032 Hungary e-mail: pongracz.andras@science.unideb.hu (see http://math.unideb.hu/pongracz-andras)

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

asymptotical behavior of the expected runtime of the three discordant protocols on the n-cycle. It is shown that the expected time for all three protocols to reach consensus on the cycle graph with n vertices satisfies $|T-\beta\varrho| = O(n^{3/2})$, where β and ϱ are the number of blue and red vertices in the initial state, respectively. In other words, on the n-cycle $T_{\text{oblivious}}, T_{\text{push}}$ and T_{pull} differ in an $O(n^{3/2})$ term, which is negligible compared to the typically quadratic runtime. The result combined with the lower estimation shows that the worst case is $\beta = \rho = n/2$, where the expected time is asymptotically $T \sim n^2/4$.

The other vital problem in case of a random protocol is to compute the probability of each outcome to win. We show that in case of the cycle graph the probability of each opinion to win with the discordant push, pull or oblivious protocol is asymptotically proportionate to the number of vertices with that opinion in the initial state, provided that there are few runs to begin with. More precisely, we demonstrate that the the blue vertices have winning probability $\frac{\beta}{n} + O(\frac{k}{n})$, where k is the number of runs in the initial state. By using some probability theory, it can be shown that there must be a state for arbitrarily large n such that the estimation $\frac{\beta}{n}$ has error greater than 0.1. However, computer simulations suggest that in highly symmetrical initial states (such as the one with alternating runs of lengths one and two), the estimation $\frac{\beta}{n}$ is quite accurate, a phenomenon we cannot explain yet.

II. Preliminaries

A. General tools

Throughout this section, P is an absorbing Markov chain with transient states T. We denote by Pen the set of potential *penultimate states* in T, that is, the states $t \in T$ such that the probability of moving from t to an absorbing state in one step is positive. As usual, we denote by Q the upper left minor of the canonical form of $P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$. So Q is the transition matrix restricted to the transient states. Following standard notations, $N = (I-Q)^{-1}$ denotes the fundamental matrix of the Markov chain. In this paper, vectors are column vectors of length $|T|$, usually denoted by $\underline{u}, \underline{v}, \underline{\varepsilon}$, etc. The coordinates are identified with the transient states, so precisely speaking, these are vectors in $\mathbb{R}^T$. We denote by $\underline{1}$ the column vector of length $|T|$ all of whose entries equal to 1. The entry corresponding to the coordinate t in the vector $\underline{u}$ is denoted by $u[t]$. It is well-known that if we sum up the entries $u[t]$ while randomly walking on the coordinates starting from $t_0 \in T$, then the expected value of this sum before the walk is absorbed is $(N\underline{u})[t_0]$. In particular, the expected times to absorption from each transient state as initial state are the coordinates of the vector $N\underline{1}$.

The following lemma is the basic observation of the elementary method we use to improve the upper estimations for the expected time to absorption presented in [17]. We can think about $x[t]$ as a "guesstimate" of the expected value of the sum of the entries of $\underline{u}$ during a random walk with initial state t before reaching an absorbing state. In particular, if $\underline{u} = 1$, then $\underline{x}$ is the guesstimate vector for the time to absorption starting from each transient state.

Elementary Lemma: Let $\underline{u}, \underline{x}, \underline{\varepsilon} \in \mathbb{R}^T$ be vectors such that $Q\underline{x} = \underline{x} - \underline{u} + \underline{\varepsilon}$. Then $N\underline{u} = \underline{x} + N\underline{\varepsilon}$. In particular, if $Q\underline{x} \leq \underline{x} - \underline{u}$, then $N\underline{u} \leq \underline{x}$ (coordinate-wise).

As we mentioned earlier, the vectors $N\underline{u}$ and $N\underline{\varepsilon}$ are the expected value vectors of the sum of the entries of $\underline{u}$ and $\underline{\varepsilon}$ during a random walk (on the coordinates) starting from each transient state. The Elementary Lemma is particularly useful when the transition matrix is large but sparse, and the fundamental matrix cannot be computed or represented in a transparent way. This is often the case with evolutionary processes. Note that $Q\underline{x}$ is easy to compute if the matrix is sparse. Furthermore, because of the probabilistic interpretation of $N\underline{\varepsilon}$ and the possibility of applying the Elementary Lemma iteratively, it is possible to estimate this vector without computing N, as we see later. By successive application of this method, the error can shrink to such a small vector that it is very easy to estimate it, providing us with an efficient estimation of the expected value vector. We spell out an immediate application.

Let $\underline{u} \in \mathbb{R}^T$ be such that $u[t] = 0$ for all $t \in T \setminus Pen$. Define $\underline{p} \in \mathbb{R}^T$ where $p[t]$ is the probability of absorption in state t. Let $M := \max_{t \in Pen} u[t]/p[t]$. Then the expected sum of the entries of $\underline{u}$ during a random walk from any initial state is at most M. (We can apply the Elementary Lemma with the guesstimate vector $M \cdot \underline{1}$.)

This observation is very advantageous when we are able to cut a process to several phases, and we want to estimate the expected sum of an expression between two phase transitions. In our case, the phases are those parts of the process where the number of *runs*, i.e., maximal intervals in the cycle that consist of vertices with the same opinion, is constant. Note that the number of runs cannot increase during the process, and it decreases by two whenever the opinion of a singleton vertex is switched. The only exception is when we reach consensus in the last step: in that case, the number of runs drops down from two to one.

B. Further terminology

We now turn to the problems under consideration, defined in the introduction. Note that the proof is briefly presented for discordant push voting on the n-cycle: the case of pull voting can be done in a similar fashion, and the case of oblivious voting is trivial. Clearly, the voting process is an absorbing Markov chain with 2^n states, whose absorbing states are exactly those two where all the vertices agree.

As in the introduction, the number of blue and red vertices are denoted by β and ϱ, respectively. A vertex is a *singleton* if its color differs from both its neighbors' color. The number of singleton blue and red vertices are s_β and s_ϱ, respectively. The number of non-singleton blue vertices with (exactly) one red neighbor is m_β; the number m_ϱ is defined analogously for red vertices. Maximal sets of consecutive vertices around the cycle with the same color are called *runs*. Note that the number of runs is even in every state, except for the two absorbing states where the whole cycle is one run. Furthermore, the number of red runs equal to the number of blue runs in the transient states, as red and blue runs alternate around the table.

III. Cutting the process

It turns out to be advantageous in the calculation to cut the process into two parts. We choose a number k_0 whose order of magnitude is $\sqrt{n}$ (in fact, $8\sqrt{5}\sqrt{n}$ is optimal for

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

estimating the runtime). The first part finishes when we reach a state with k_0 runs.

A. The first part: down to $O(\sqrt{n})$ runs

In this subsection, we show an estimation of the expected length of the first part. The following bound can be extracted from [17, Section 4]. In that paper, a quadratic upper estimation was given to the runtime of the discordant push protocol using some results about stopped martingales. They obtained that it takes at most $33n^2$ steps to reach consensus from any initial state, that is, to reach a state with one run. However, by carefully modifying their calculations, it can be shown that the expected time to reach a state with k runs is at most $80n^2/k$ from any initial state. Thus if $k_0 = \Theta(\sqrt{n})$, then the first part is expected to terminate in $O(n^{3/2})$ steps.

In [17, Lemma 8] and the argument before that, it was shown that the expected time to reach a state with $k = 2r_1$ runs from one with $2r_0$ runs is at most T^*, where T^* is the optimal solution of the following linear program:

$$T^* = \max 10\sqrt{2}n^{3/2} \sum_{r=r_1}^{r_0} \frac{x_r}{r^{3/2}}$$

$$\text{such that} \quad \sum_{j=r_1}^{r} x_j \leq \sqrt{2rn} \quad \text{for all} \quad r_1 \leq r \leq r_0$$

$$\text{and} \quad x_r \geq 0 \quad \text{for all} \quad r_1 \leq r \leq r_0$$

Moreover, it can be shown that such a linear program attains its optimal solution at $x_{r_1} = \sqrt{2r_1 n}$ and $x_r = \sqrt{2rn} - \sqrt{2(r-1)n}$ for all $r_1 + 1 \leq j \leq r_0$. Hence, by using standard estimations we obtain $T^* \leq \frac{40n^2}{r_1} = \frac{80n^2}{k}$.

B. The second part: from $O(\sqrt{n})$ runs to consensus

As we suggested earlier, it seems impossible to compute the fundamental matrix of our Markov chain. However, the upper-left minor Q of the transition matrix is sparse, so the Elementary Lemma and its consequences can be applied.

The way we phrased the result in the introduction provides the right heuristics for the guesstimate vector. The expected runtime of the oblivious protocol is clearly $\beta\varrho$: it is simply the runtime of a drunkard walk with parameter $n = \beta + \varrho$ and initial state β (see [17] for details). Computer simulations (in SAGE) suggested that the runtime of the three discordant protocols should be close to each other, but it is not so easy to turn this intuition into a precise proof. That is why we use the Elementary Lemma with guesstimate vector $\underline{x}$ whose entries are $\beta\varrho$ for each transient state (where β and ϱ depends on the state).

The probability of the number of blue vertices to increase by 1, i.e., a blue vertex is pushing, is

$$\frac{s_\beta + m_\beta}{s_\beta + m_\beta + s_\varrho + m_\varrho}$$

Similarly, the probability of the number of red vertices to increase by 1 is $(s_\varrho + m_\varrho)/(s_\beta + m_\beta + s_\varrho + m_\varrho)$. If we multiply the value of the vector $\underline{x}$ at those states with the transition probabilities, and add them up, i.e., we calculate $Q\underline{x}$, we obtain that the error vector in the Elementary Lemma is $\frac{(\varrho-\beta)(s_\beta+m_\beta-s_\varrho-m_\varrho)}{s_\beta+m_\beta+s_\varrho+m_\varrho}$. In this expression, $|\varrho - \beta| \leq n$, so it is enough to estimate the sum of $|\frac{s_\beta+m_\beta-s_\varrho-m_\varrho}{s_\beta+m_\beta+s_\varrho+m_\varrho}|$ during a random walk.

It can be shown by the above presented corollary of the Elementary Lemma that the expected value of the sum of $|\frac{s_\beta+m_\beta-s_\varrho-m_\varrho}{s_\beta+m_\beta+s_\varrho+m_\varrho}|$ during a random walk between two phase transitions (while the number of runs is constant) is at most $1/2$. We omit the complicated combinatorial argument.

In particular, the expected sum of the above expression during the second part of the voting process is at most $O(\sqrt{n})$, as there are $O(\sqrt{n})$ phase transitions from a state with $O(\sqrt{n})$ runs.

Thus by adding up these estimations in the first and second parts of the process, we obtain that the expected time of reaching consensus is

$$O(n^{3/2}) + (\beta\varrho + O(n^{3/2})) = \beta\varrho + O(n^{3/2})$$

.

IV. Winning probabilities

It is enough to estimate the winning probability p of the color blue, the other color then wins with probability $1 - p$. Again, we know from standard theory that the matrix NR consists of the probabilities to reach from transient state i the absorbing state j in the process. So we are only interested in the first column of this $(2^n - 2) \times 2$ matrix. The Elementary Lemma can be applied, as the problem is to estimate the vector $N\underline{u}$ where $\underline{u}$ is the first column of R. The calculation is similar to the estimation of the expected runtime, and in fact, a very similar error term is obtained using the guesstimate vector with entries $\frac{\beta}{n}$, namely $\frac{s_\beta+m_\beta-s_\varrho-m_\varrho}{n(s_\beta+m_\beta+s_\varrho+m_\varrho)}$. As there are $k/2$ phase transitions from a state with k runs until reaching consensus, and the expected value of the sum $\frac{s_\beta+m_\beta-s_\varrho-m_\varrho}{s_\beta+m_\beta+s_\varrho+m_\varrho}$ is $1/2$, we obtain the estimation $|p - \frac{\beta}{n}| = O(\frac{k}{n})$.

V. Further results and future work

It is also possible to obtain asymptotically sharp estimates for the corresponding problems in the star graph with n vertices. This is a typical network, when one server is connected to several clients. It was already pointed out in [17] that, quite counter-intuitively, the discordant pull protocol is faster than the discordant push protocol on such graphs if n is large enough. We were able to refine this result, and obtain asymptotically sharp estimations for both runtimes. The author also managed to show that the three discordant protocols presented in the introduction differ from their linear counterparts in terms of winning probabilities by an error whose order of magnitude is $o(1)$, if the underlying graph is the random graph $G(n, p)$. The most intriguing open problem is the one mentioned in the introduction. The proof that the estimation $\frac{\beta}{n}$ must have an error at least 0.1 on some initial state of the cycle graph strongly hints at the initial state whose runs have alternating lengths one and two. However, computer simulations suggest that in this particular case the winning probability for the blue vertices converges to $\frac{2}{3}$ as n tends to infinity. Thanks to the $O(n^{3/2})$ estimation to the runtime of the first part, it is possible to estimate the desired probability up to an error term $O(1/\sqrt{n})$ if one is willing to run the process for $O(n^{3/2})$ steps. This is a good trade-off, as the expected time to reach consensus from the worst initial case is quadratic. In fact, using this observation, it is possible to write a relatively fast program that runs the discordant

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

push protocol on the cycle with 5000 vertices 5000 times from the initial state above. If this empirical result is correct, then there must be a more complicated formula that takes into consideration the position of blue vertices around the cycle as well as their number, and coincidentally this formula should assign $\frac{2}{3}$ to the above mentioned initial state.

REFERENCES

[1] P. Donnelly and D. Welsh, "Finite particle systems and infection models," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 94, no. 1, pp. 167–182, 1983.

[2] T. Nakata, H. Imahayashi, and M. Yamashita, *Probabilistic local majority voting for the agreement problem on finite graphs*. Springer, 1999.

[3] R. Karp, C. Schindelhauer, S. Shenke, and B. Vocking, "Randomized rumor spreading," in *FOCS '00 Proceedings of the 41st Annual Symposium on Foundations of Computer Science 2000*, pp. 565–574.

[4] H. Acan, A. Collevecchio, A. Mehrabian, and N. Wormald, "On the push&pull protocol for rumour spreading," in *Proceedings of the ACM Symposium on Principles of Distributed Computing 2015*, pp. 405–412.

[5] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer, "Push-to-pull peer-to-peer live streaming," in *Proceedings of the International Symposium on Distributed Computing 2007*, pp. 388–402.

[6] R. Durrett, J. P. Gleeson, A. L. Lloyd, P. J. Mucha, F. Shi, and D. Sivakoff, "Graph fission in an evolving voter model," *Proceedings of the National Academy of Sciences*, vol. 109, no. 10, pp. 3682–3687, 2012.

[7] Y. Hassin and D. Peleg, "Distributed probabilistic polling and applications to proportionate agreement," *Information and Computation*, vol. 171, no. 2, pp. 248–268, 2001.

[8] P. Holme and M. E. J. Newman, "Nonequilibrium phase transition in the coevolution of networks and opinions," *Physical Review E*, vol. 74, no. 5, p. 5 pp, 2006.

[9] R. Basu and A. Sly, "Evolving voter model on dense random graphs," *Annals of Applied Probability*, vol. 27, no. 2, pp. 1235–1288, 2017.

[10] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.

[11] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 121–133, 2004.

[12] D. Carra, R. L. Cigno, and A. Russo, "On some fundamental properties of p2p push/pull protocols," in *Proceedings of the Second International Conference on Communications and Electronics 2008*, p. 7 pp.

[13] R. Oliveira, "On the coalescence time of reversible random walks," *Transactions of the American Mathematical Society*, vol. 364, no. 4, pp. 2109–2128, 2012.

[14] ——, "Mean field conditions for coalescing random walks," *The Annals of Probability*, vol. 41, no. 5, pp. 3420–3461, 2013.

[15] C. Cooper, R. Elsasser, H. Ono, and T. Radzik, "Coalescing random walks and voting on connected graphs," *SIAM Journal on Discrete Mathematics*, vol. 27, no. 4, pp. 1748–1758, 2013.

[16] C. Cooper and N. Rivera, "The linear voting model," in *43rd International Colloquium on Automata, Languages, and Programming*, ICALP 2016, pp. 2021–2032.

[17] C. Cooper, M. Dyer, A. Frieze, and N. Rivera, "Discordant voting processes on finite graphs," in *43rd International Colloquium on Automata, Languages, and Programming*, ICALP 2016, pp. 2033–2045.

ISBN: 978-988-14047-9-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)