Applied Network Science

RESEARCH Open Access

Engineering consensus in static networks with unknown disruptors

Agathe Bouis[1*], Christopher Lowe[1], Ruaridh A. Clark[1] and Malcolm Macdonald[1]

*Correspondence: agathe.bouis@strath.ac.uk

[1] Centre for Signal and Image Processing (CeSIP), Department of Electric and Electrical Engineering, University of Strathclyde, Glasgow, Scotland

Abstract

Distributed control can increase system scalability, flexibility, and redundancy. Foundational to such scalability via decentralisation is consensus formation, by which decision-making and coordination are achieved. However, decentralised multi-agent systems are inherently vulnerable to disruption. To develop a resilient consensus approach, inspiration is taken from the study of social dynamics; specifically, the Deffuant Model which evaluates the impact of tolerance in social systems. A dynamic protocol is presented enabling efficient consensus to be reached with an unknown number of disruptors present within a multi-agent system. By inverting typical social tolerance, agents filter out extremist non-standard opinions that would drive them away from consensus. This approach allows distributed systems to deal with unknown disruptions, without knowledge of the network topology or the numbers and behaviours of the disruptors, a general requirement of other resilient consensus algorithms. A disruptor-agnostic algorithm is particularly suitable to real-world applications where information regarding disruptors or network properties is typically unknown. Faster, tighter, and more robust convergence can be achieved across a range of scenarios with the social dynamics inspired algorithm presented herein, when compared with Mean-Subsequence-Reduced-type methods.

Keywords: Consensus, Fault-tolerance, Disruption-tolerance, Social dynamics, Multi-agent system, Distributed, Resilience

Introduction

Methods of network control are shifting from centralised to distributed to avoid communication overheads, their associated latencies, and single point failures (Ding et al. 2018; Sakavalas and Tseng 2019). Distributed control strategies must cope with issues of security and resilience to faults, and deliberate attacks against ever growing networks (both in number of nodes and connections). Although such issues are also present in centralised systems, distributed networks prove to be particularly vulnerable as they lack a central authority to authenticate nodes as well as monitor and detect misbehaviours or failures (Michiardi and Molva 2002). Decentralised control methods are thus characterised by the realistic systems to which they are applied: systems with unknown numbers of disruptors and unknown network topologies. Typical networks include the likes of Internet of Things, mobile robotic systems, smart power grids, and wireless sensor networks.

The foundation of distributed control is consensus, whereby nodes within a system reach a common shared state (Olfati-Saber et al. 2007). This is often achieved using linear consensus iterations, where at each time step network nodes update their state values based on a weighed combination of their neighbours' values (Pasqualetti et al. 2011; Jadbabaie et al. 2003). The choice of combination is what decides on a consensus' disruption tolerance. Disruption tolerance is commonly defined as the ability to cope with, and/or recover from disruptions within the network (Al-Kuwaiti et al. 2006). These malfunctions can be faults or deliberate attacks (LeBlanc et al. 2013), which are manifested in this paper under the same form: undetectable disruptions. The simplest approach to selecting which neighbouring nodes to combine is mean-based computation; where the average of all neighbours' values is used for the state update. However, these types of methods are not resilient as nodes can be swayed by the presence of disruptors (LeBlanc et al. 2013). They must therefore be adapted to ensure fault-resilience; usually by filtering incoming neighbour state values and/or by augmenting the system in some way such as introducing virtual connectivities to collect more values on which to base the update (Phan and Kim 2020), or implementing observers to create adaptive controllers (Lv et al. 2019; Trejo et al. 2020). One such typical approach is the Mean-Subsequence-Reduced (MSR)-type method (Kieckhafer and Azadmanesh 1994), and its extensions (e.g. Weighted-MSR (LeBlanc et al. 2013; Su and Vaidya 2017), Omission-MSR (Azadmanesh and Kieckhafer 2000). MSR methods ensure their fault-tolerance by removing a set number of the largest and smallest values received by nodes at each consensus step, enabling agents to reach an agreement despite disruptive values. The removed numbers correspond to the maximum possible number of faults that the system can cope with (Zhang et al. 2012). By requiring knowledge of this design value, MSR method are observed to respond poorly to situations where this information is unavailable as well as lacking robustness to additional failures.

To provide a dynamic and robust solution to the consensus problem, requiring neither information regarding the topology nor indication of the number of disruptive agents in the system, this paper presents a fully decentralised, adaptive consensus algorithm handling the specifications of realistic, static networks using a framework adapted from the dynamics of social systems. Specifically, a sociophysics approach is adopted. Sociophysics models the behaviour of human crowds and the social interactions therein (Castellano et al. 2009; Weisbuch et al. 2002). That is, people in everyday life influence one another and adapt towards, or away from, the expressed opinions of the people they have interacted with.

A model of interest, notably regarding the spreading of minority opinion (Xie et al. 2011), is that proposed by Deffuant et al. (2006). Deffuant's Bounded confidence (BC) is a stochastic model of the evolution of continuous-valued opinions within a given range (Mathias et al. 2016), usually set between -1 and 1 (Deffuant et al. 2002); moderate opinions close to 0 and extreme ones approaching either -1 or 1. The framework can be understood as: two agents coming into contact must hold sufficiently close opinions in order to mutually influence one another. When encountering agents of radically different opinions, that is, when the difference of opinions is too large, there are no changes with respect to the opinions held prior to the communication. The threshold for the difference is known as the agent's tolerance, d. Denoting the opinion of agent i at a given time t as $o_i[t]$, the interactions of agent i with agent j can be described as

$$o_i[t+1] = o_i[t] + \eta\left(o_j[t] - o_i[t]\right) \text{ if } \left|o_j[t] - o_i[t]\right| < d \tag{1}$$

In Deffuant social dynamics, an agent will typically only interact with one other agent at a time, with their conversation partner j being drawn at random from the pool of other agents. If a scenario is considered where agents have multiple fixed connections to different neighbours then it holds that the opinion of an agent i will be influenced by multiple other agents such that $o_i[t+1]$ will be the result of multiple interactions with j as a part of the set of agents connected to i.

When implementing a fixed universal tolerance, the final distribution of opinions can be categorised as; consensus, when a single opinion dominates; polarisation, when two opinions emerge; and, fragmentation, when several opinion clusters can be observed (Phan and Kim 2020). The transition from one phase to the next depends on the universal tolerance threshold.

A secondary dimension to improve the model's realism was introduced by Sobkowicz (2015). By deriving an explicit correlation between extreme opinions and low tolerance, the tendency for extremists to be inflexible is explained. It also illustrates the propensity of moderate agents to be more open minded and thus easily swayed towards extremism (Sobkowicz 2015; Guarino et al. 2020; Araque and Iglesias 2020). This can be exploited by bad actors to lead to dissention, polarisation, and fragmentation of opinions (Guarino et al. 2020; Berghel 2018; Pierri et al. 2020). This behaviour is addressed by morphing the universal tolerance, d, into individual agent's tolerance. These tolerances are then updated as a function of the agent's own opinion with respect to other opinions as $d_i = f(|o_i|)$.

The mechanism of belief diffusion is not solely observed in opinion dynamics, it also represents the foundations of consensus algorithms in network wherein agents communicate with their (usually one-hop) neighbours to update their own state variables. The source of conflict in both model and algorithm can be understood to stem from similarly disruptive attitudes. As observed by the rise in political extremism propelled by small but vocal minorities, in social environments agents with extreme opinions compel other non-extreme agents (moderates) into becoming extremists themselves (Castellano et al. 2009; Xie et al. 2011). This is also the case for engineered networks, where the presence of even small minorities of disruptive, selfish, or malicious agents can come to prevent consensus (Michiardi and Molva 2002).

This paper exploits, and inverts, the mechanisms of opinion diffusion to design a consensus algorithm resistant to faulty/disruptive actors (extremists). The algorithm, termed the Opinion Dynamics-inspired DIsruption- tolerant Consensus, ODDI-C (pronounced *o·duh·see*), enables a resilient consensus to be reached despite a lack of awareness of network topology and number of disruptive agents. This approach is chosen based on the similarity between how agents in bounded confidence social models and nodes in linear consensus scenarios update their opinions/values. On the basis of these similar mechanisms of opinion/state value update, features of social systems resulting in consensus are analysed, extracted, and implemented in the context of a consensus algorithm. In practice, this is achieved through the exploitation of a dynamic tolerance linked to extremism, whereby agents filter out extreme non-standard opinions driving them away from consensus. This make the approach particularly suitable to real-world applications where there is limited, to no, external information and where synchronisation is critical.

Opinion dynamics-inspired disruption-tolerant consensus algorithm

The ODDI-C Algorithm adapts the concept of tolerance from social systems, using an analogous approach to the Deffuant Model for its update and filtering. That is, an agent's tolerance increases when it finds itself more extreme, and all values beyond an agent's tolerance are filtered out and not considered during the update. Moreover, the ODDI-C Algorithm deals with the median-based z-scores rather than raw opinions. The median-based z-score quantifies how close an opinion is to the local median of the distribution. The replacement of the raw opinion with the z-score serves two purposes. The first is to allow for any range of input to be considered, thus expanding the typical Deffuant model's range of -1 to 1. Secondly, and more importantly, the use of the z-score serves as inbuilt tolerance indicator.

The consensus of compliant agents can be ensured by having them only consider opinions which are closer to the median than they are. The z-score facilitates this by acting as a dynamic tolerance for the agent doing the filtering, enabling it to only consider opinions whose z-score are smaller than its own. That is, agents standing close to the median of the distribution will prove particularly stubborn while agents who are outlier themselves will be much more willing to account for other agents' opinions. This is an inversion of typical social systems, for whom extremism is linked to low tolerances. This approach also provides an elegant solution to the problem of filtering out extreme values while allowing for a level of adaptability dependent on the agent's own opinion.

Note herein the use of the median rather than the mean when dealing with z-scores due to its statistical robustness to outliers. The median is more stable than the mean when dealing with heavy tailed or asymmetric distributions, and as such, it is a more appropriate tool for outlier filtering (Leys et al. 2013). Adapted from Raynal (2002), three consensus properties act as algorithm design requirements.

- Termination: All compliant agents must decide on a value.
- Validity: The agreed upon value is not influenced or dictated by disruptive agents.
- Agreement: No two compliant agents decide on different values, to within an agreed tolerance.

Note that Termination does not require Agreement, in which case polarisation or fragmentation are obtained.

The ODDI-C Algorithm is primed for systems dealing with faults/attacks of unknown identities and behaviours Considering this unknown, the validity property is impossible to ensure (Civit et al. 2023). The property is thus weakened, following (Civit et al. 2023), to be applicable for such scenarios, which use linear consensus mechanics with unknown disruptors.

- Weak Validity: If all nodes are compliant and propose the same value, all nodes will agree on this value.

Algorithm rules

The algorithm assumes a network constructed as follows: Let $G = (V, \varepsilon, A)$ be a digraph with a finite number of nodes (agents) $V = 1, 2, ..., N$, a set of directed edges $\varepsilon \subseteq V \times V$, and an adjacency matrix $A = a_{ij} \in R(\mathbb{N} \times \mathbb{N})$. Each directed edge $(j, i) \in \varepsilon$ represents a directional link between the node pair (j, i), such that communication between the nodes is enabled. In the model described here, it is assumed that the nodes communicate synchronously at each given opportunity such that the edges model the flow of information between the two nodes. A node can therefore be influenced by multiple neighbours at the same time step. The set of nodes influencing node i, it's in-neighbours, is defined $V_i^{in} = j \in V : (j, i) \in \varepsilon$ while the nodes influenced by i, its out-neighbours, are in the set $V_{\mathrm{i}}^{out} = j \in V : (i, j) \in \varepsilon$. The components of the adjacency matrix $a_{ij} = 1$ if $j \in V_i$, that is, if node i can receive information from node j, otherwise $a_{ij} = 0$. It should also be noted that each node has access to its own value at each time step, thus making it consider its inclusive neighbourhood.

The synchronous ODDI-C Algorithm implemented by compliant agents is presented in Fig. 1.

The algorithm's filtering process works as follows, starting with Neighbour State Value Collection (see Fig. 1). At each time step $t \in N$, each compliant node i sends its state variable $\psi_i[t]$ to all of the nodes in its out-neighbourhood $j \in V^{out}$ out and in turn receives the state variables $\psi_j[t]$ from its in-neighbours $j \in V^{in}$ (compliant and disruptive, if they exist). These values are known as $\Psi_i[t]$,

$$\Psi_i[\mathrm{t}] = \left\{ \psi_j[t] : j \in V_i^{in} \right\} \tag{2}$$

From these received values, the Filter Parameter Creation can begin. These parameters are the median $\tilde{\Psi}_i^{\Psi_i}[t]$ and the Median Absolute Deviation $\mathrm{MAD}_i^{\Psi i}[t]$ (a more robust alternative to the standard deviation) of the concatenated array $\Psi_i[t]$ and are determined from

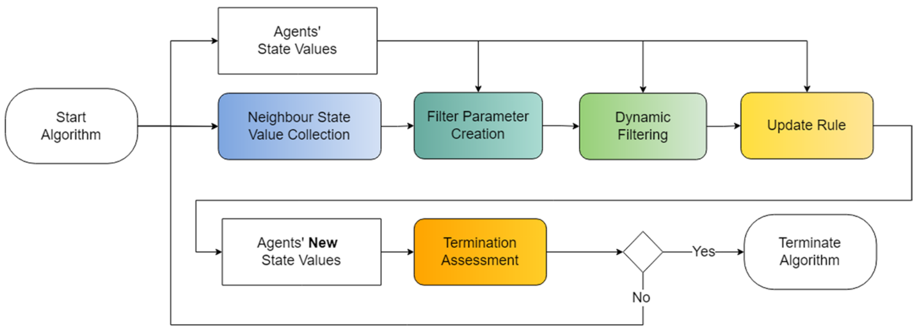

Fig. 1 ODDI-C Algorithm Flow Diagram. The algorithm is repeated for each compliant node at each time step until the termination criterion (or criteria) is (are) met

$$\tilde{\Psi}_i^{\Psi_i}[t] = \text{median}(\Psi_i[t]) \tag{3}$$

$$\text{MAD}_i^{\Psi i}[t] = \text{median}\left(\left|\Psi_i[t] - \tilde{\Psi}_i^{\Psi_i}[t]\right|\right) \tag{4}$$

When dealing with normal distributions, the MAD can be scaled using a factor of 1.4826 (Leys et al. 2013; Rousseeuw and Croux 1993). The use of the MAD and its normalised scaling,

$$\text{NMAD}_i^{\Psi i}[t] = 1.4826 \cdot \text{MAD}_i^{\Psi i}[t], \tag{5}$$

relies on the assumption of a normal, mostly symmetric distribution of node values. For other distributions, other estimators or scalings may be used as they may be more appropriate (Rousseeuw and Croux 1993). Distribution here refers to the dominant distribution of node values, meaning that unless the number of disruptive agents dominates the network, the distribution will be that of the compliant agents. When simulating this consensus problem, the distribution of the node's initial values is used to assess which scaling to use. This in turn allows for the median-based z-scores of the received values to be calculated. These values are kept as absolutes as the sign is of no importance for the filtering. The z-scores of the received values is determined as,

$$\text{z - score}_i(\Psi_i[t])[t] = \left(\frac{\left|\Psi_i[t] - \tilde{\Psi}_i^{\Psi_i}[t]\right|}{\text{NMAD}_i^{\Psi i}[t]}\right), \tag{6}$$

using the median ($\tilde{\Psi}_i^{\Psi_i}[t]$) and scaled MAD ($\text{MAD}_i^{\Psi i}[t]$) of the distribution. The filter value,

$$\text{filter}_i[t] = \left(\frac{\left|\psi_\text{i}[\text{t}] - \tilde{\Psi}_i^{\Psi_i}[t]\right|}{\text{NMAD}_i^{\Psi i}[t]}\right) = \text{z - score}_i^i, \tag{7}$$

is then created based on the z-score of the agent (and its opinion $\psi_i[t]$).

Before the filter can be used, it must be optimised. This is accomplished through the implementation of dynamic upper limit. The method used is a Hampel filter, a method typically applied to limit the classification of nodes as outlier/not-outlier. Nodes are classified as outliers if they are larger/smaller than 3 scaled Median Absolute Deviations (NMAD) (or other MAD scalings if the normal distribution assumption were disapplied) (Leys et al. 2013). Considering our use of z-score as discriminant (rather than the raw state value), the Hampel filter is adapted as,

$$\text{filter}_{max}[t] = 3 \tag{8}$$

$$\text{filter}_i[\text{t}] = \min\left\{\text{z - score}_i^i, \text{filter}_{max}[t]\right\} \tag{9}$$

Once the filter value is set for each agent, they can then proceed to the Dynamic Filtering, assessing which values will be used to contribute to the update. Values are filtered out if their z-scores are higher than the dynamic filter value. Selected values ($F_i[t]$) can be understood as

$$\mathrm{F}_i[\mathrm{t}] = \left\{ \Psi_i[t] \mid \text{z - score}_i(\Psi_i[t])[t] < \text{filter}_i[\mathrm{t}] \right\} \tag{10}$$

meaning that the only values the node will accept will be closer to the median (centre of the distribution) than its own value.

Following the filtering, agents will follow the Update Rule to update their state values, ($\psi_i[t+1]$), based on the average difference between the selected values ($f_j[t] \in F_i[t], j = 1, ..., |F_i|$) and its own past value, ($\psi_i[t]$). The extent of the update will be tuned by the learning rate of the consensus, η. Small learning rates require more update steps and may result in the process seeing a reduced convergence rate, or getting frozen. On the other hand, too large a learning rate may result in the nodes failing to converge. This parameter, which is typically set between [0, 1], can be tuned for improved performance (Murphy 2012). Here, it is set to be 0.5 as a middle ground to avoid over- or underdamping the system. The resulting new state value is expressed as

$$\psi_i[t+1] = \psi_i[t] + \eta \cdot \frac{1}{|F_i|} \sum_{j=1}^{|F_i|} \left(\psi_i[t] - f_j[t] \right) \tag{11}$$

The algorithm is repeated for each compliant node and for each time step until the Termination Assessment is met. This assessment's criterion can be a maximum time, where the simulation stops once maximum time is exceeded ($t > t_{max}$), or a value-based decision, where the simulation stops once convergence is within acceptable threshold.

Convergence metric

To visualize the evolution of compliant nodes' opinions over time, a simple metric assessing convergence is implemented. Convergence is the process by which the agreement and termination properties are met. The Convergence Metric (CM) is calculated at each time step as the Total Difference (TD) between compliant agents (where C is the set of compliant agents excluding disruptive nodes), normalised with respect to the initial difference between all nodes ($TD[0]$),

$$\mathrm{CM}[t] = \frac{\mathrm{TD}[t]}{\mathrm{TD}[0]} \tag{12}$$

$$\mathrm{TD}[t] = \sum_{k=1}^{C} \sum_{m=1}^{C} \left(|\psi_k[t] - \psi_m[t]| \right) \tag{13}$$

where, ψ_k are compliant agents' opinions $\psi_k \in \Psi_k$ for $k \in C$ and ψ_m are agents' opinions $\psi_m \in \Psi_m$ for $m \in C$ such that between the two loops k and m, all compliant agent opinions are compared, and their absolute difference tabulated. The relative metric decreases when nodes converge, agreeing on a shared state, and increases when nodes diverge. Note that the metric can never be negative as the TD is calculated from the absolute difference between the nodes.

The convergence metric is a relative value dependent on the initial difference between nodes' values ($TD[0]$). As a result, the exact difference observed between the nodes can be difficult to understand on its own, even when contextualised by the plot of nodes' opinion trajectories. The same value of convergence metric can be the same for two different simulation, but will correspond to two different absolute differences dependent on the initial difference of each of the simulation. To allow for simulations to be compared and to avoid rounding errors as the difference between nodes becomes too small (or even zero), an error threshold (*err*) is applied. This threshold is used to create a minimum floor value (*Floor*) for the convergence metric. Results smaller than this floor are levelled to the floor value. This error threshold corresponds to a small absolute difference between the nodes, which is then translated to a relative floor dependent on the initial difference. This adaptation of the convergence metric prevents the development of small rounding errors while giving context to the results.

$$\text{Floor} = \frac{err}{\text{TD}\ [0]} \tag{14}$$

$$\text{if} \qquad \text{CM} < \text{Floor} \qquad \text{then} \qquad \text{CM} = \text{Floor} \tag{15}$$

the error threshold (*err*) is set at a value of 10^{-7}.

Results

Four numerical experiments are used to investigate the algorithm's performance. Experiment 1 considers a single network with disruptors, where the performance of the ODDI-C Algorithm is compared against a scenario with a classical mean-based algorithm (no filtering) implemented. This allows for the confirmation of the Weak Validity property and for the behaviour of compliant nodes following ODDI-C to be examined in the face of specific patterns of disruptive behaviours. Experiment 2 assesses the performance impact of changing network connectivity in the face of a constant number of disruptions using a series of small Monte Carlo simulation (50 runs) per connectivity level. This test enables for ODDI-C's performance to be measured for various connectivity levels, assessing the connectivity's impact on the Termination and Agreement properties. ODDI-C's output is compared against the MSR-Algorithm's (LeBlanc et al. 2013) for the same networks, its behaviour being a point of reference in the field of disruption tolerant consensus (details of its implementation are provided in Section: MSR Implementation).

Experiment 3 uses a sequence of small Monte Carlo simulations to evaluate ODDI-C's performance degradation (deterioration of compliance with the Termination and Agreement properties) when dealing with increasing numbers of disruptions for networks with fixed connectivity. This test is used to investigate the Termination and Agreement properties of the algorithm, particularly regarding ($|D|$), the number of disruptors, which overwhelm attempts at consensus building. Similar, to Experiment 2, each simulation is duplicated using MSR as comparison. Experiments 2 and 3 are both performed on 20-node networks to allow for a sufficient number of message exchanges and nodes to showcase the algorithm's scaling while remaining of a comprehensible and manageable size.

The premise of ODDI-C Algorithm is an undetectable fault or attack scenario attempting to disrupt the system's ability to achieve Termination and Agreement (Ferrari and Teixeira 2021).. Disruptive agents are understood as being "deceptive", with non-disruptive agents remaining unaware of their presence in the system. Disruptive agents do not follow the ODDI-C Algorithm, instead adopting unknown behaviours and/or values and broadcasting them (Shang 2021). Three types of disruptions are used across the three Experiments: a sine wave (T1, described in Eq. 16), a linearly increasing function (T2, described in Eq. 17), and a noise pattern (T3, described in Eq. 18). The functions describing them are

$$\mathrm{T1}: \left(\overline{\Psi}[0] + s_y\right) + (A \cdot r) \cdot \sin\left(\omega t + s_x\right) \tag{16}$$

$$\mathrm{T2}: \overline{\Psi}[0] + m \cdot t \tag{17}$$

$$\mathrm{T3}: \overline{\Psi}[0] + \text{ rand} \cdot 0.9 \cdot r \tag{18}$$

where $\overline{\Psi}[0]$ is the average of all nodes (including ones that will become disruptive) at time 0, s_y is the vertical y-axis shift of the sine wave disruption, s_x is the horizontal x-axis shift of the sine wave disruption, A is the amplitude factor of the sine wave disruption, r is the range of node values at time 0, ω is the angular frequency of the sine wave disruption, m is the gradient of the linear function, and rand is a uniformly distributed random number in the interval $[0, 1]$. Initial node values of are drawn from normal distributions with mean μ and standard deviation σ (this distribution allows for the scaling of the MAD to NMAD to hold, as per Eq. 5), and described as

$$\Psi[0] \sim \mathcal{N}\left(\mu, \sigma^2\right) \tag{19}$$

In all test cases, disruptive agents become active from time step 2 onward. These nodes keep the same identity throughout the run. The nodes do not recover and are considered to remain disruptive for the extent of each run. The convergence metric is determined and used to compare ODDI-C and MSR's behaviour in multi-run scenarios. The termination criterion for all Experiments is a maximum number of timesteps such that $t_{max} = 40$, with the simulation stopping once this maximum time is reached.

For Experiment 1, the network robustness is provided. This metric is provided to enable comparisons of the proposed algorithm performance to other consensus algorithms, which test their algorithms on graphs described by their robustness. Networks are changed randomly for Experiments 2 and 3, as such their robustness are not provided considering their lack of relevance. As defined by LeBlanc (LeBlanc et al. 2013), network robustness formalises the notion of redundancy of direct information exchange between the various subsets in the network. It is a term which captures the idea that there are enough nodes in every pair of non-disjoint sets within the network that have at least a given number of neighbours outside of their respective sets. That is, that each node receives enough information so that it will not be swayed by disruptive agents attempting to "contaminate" the subset to which the node belongs. The reader is directed towards (LeBlanc et al. 2013) for details regarding this robustness calculation.

MSR implementation

MSR relies on knowledge of the number of disruptions, $|D|$, in the system, with at most $|D| \times 2$ being filtered out. At each time step, each node i receives its in-neighbours' values, $\Psi_i[t]$, forms a sorted list, and proceeds to the filtering. If there are less than $|D|$ values larger than the node's own value ($|D| > |\Psi_i[t] > \psi_\mathrm{i}[\mathrm{t}]|$), then node i only removes the values strictly larger than its own ($|\Psi_i[t] > \psi_\mathrm{i}[\mathrm{t}]|$), otherwise it removes precisely the number of disruptive values known to be present in the system ($|D|$). Likewise, if there are less than $|D|$ values strictly smaller than the node's own value ($|D| > |\Psi_i[t] < \psi_\mathrm{i}[\mathrm{t}]|$), only this number will be removed. Otherwise, precisely $|D|$ smaller than the node's values will be removed. The remaining values $F_i[t]$ are used in linear consensus update as described in Eq. 11. MSR's implementation in Experiments 2 and 3 follows (LeBlanc et al. 2013).

MSR was selected for comparative analysis with ODDI-C due to its ubiquitous nature in the field of resilient consensus for multi-agent systems (Ishii et al. 2022). Multiple MSR methods were tested and assessed, with the general version of MSR being preferred over other, more specific, MSR methods due to its generality, applicability, and overall better performance.

The general MSR method disregards a varying number of nodes, with up to $|D|$ values being filtered out. Simulations were performed using alternative MSR filters consistently disregarding a fixed number of $|D|$ nodes (Protocol 1 for the M1 Model (Wang et al. 2021)). However, results proved to be poorer than the general MSR and were thus disregarded.

The Trusted-Region Subsequence Reduction (TRSR) (Zhai et al. 2020; Abbas et al. 2017), another modification of MSR, was also investigated. With this method, the selection of the parameter $|D|$ is avoided by designing a trusted region for each node. This region is created by designating a small subset of compliant agent as trusted agents, whose behaviour is certified to be non-disruptive. Compliant agents know which information is sent by trusted agents and filter out all values they receive which are outwith the minimum and maximum bounds set by the trusted agents. Simulations performed with TRSR (using two trusted nodes per compliant node) outperformed both MSR and ODDI-C when dealing with large disruptive to compliant node ratios, being able to effectively ignore large numbers of disruptive nodes. However, in cases with few disruptive nodes and/or high connectivity, TRSR over filtered incoming values and proved to have a much slower rate of convergence than either MSR or ODDI-C (especially for low number of disruptive agents for which TSR over filtered incoming data). Furthermore, some tests could not be performed with TRSR due to it requiring at least two trusted connections per each compliant node. For example, simulations with low network connectivity where the intersection of the sets of compliant nodes and connected nodes was often too small to allow the selection of two trusted nodes, leading to simulation failures. Additionally, while this technique has shown itself to outperform the general MSR, the need for the system to rely on known trusted nodes reduces its applicability in a fully decentralised scenario such as the one envisioned for ODDI-C. In that respect, TRSR is more akin to a leader–follower resilient consensus where the security of certain agents (and their datalinks) can be ensured.

ODDI-C is a general, purpose- and network-agnostic consensus protocol designed to provide resilience against non-specific disruptions (failures, malfunctions, and malicious nodes). Such a scenario precludes protocols requiring additional information regarding the network/node/link properties (Coretti et al. 2022; Renganathan et al. 2017; Senejohnny et al. 2019; Saldana et al. 2017; Yuan et al. 2022) or which imply a loss of generality of the nodes' disruptiveness by requiring for a difference to be made between malfunctions and malicious behaviour (Wang et al. 2021; Yuan et al. 2022; Garay 1994). As a result, more specialised MSR-type and epidemics methods were found to be inapplicable and were discarded from the comparison.

Experiment 1: single run

Two multi agent networks are assessed, one with $N = 7$ nodes, the other with $N = 15$ nodes. In the first case, the disruptive agents are $D = 1, 7$, in the 15-node case they are $D = 4, 12$. The 7-node network is (3,3)-robust while the 15-node is (5,1)-robust (in addition to being (4,2)- and (3,2)-robust). All agents (compliant and disruptive) have initial values drawn from a normal distribution. The 7-node case have initial conditions between 10 and 100 (distribution described in Eq. 19, parameters $\mu = 50.5$, $\sigma = 24.75$) and deals with two disruptive sine waves (T1, as described in Eq. 16), one with a large period and amplitude, and another with a small period and amplitude. The 15-node case has initial conditions set between 1 and 15 (distribution described in Eq. 19, parameters $\mu = 8.$, $\sigma = 7$) and its disruptive behaviours are a linearly increasing function (T2, as described in Eq. 17) and a noise function (T3, as described in Eq. 18). The parameters describing the disruptive behaviours in Experiment 1 are shown in Table 1 for the 7-node and 15-node cases.

The results of the simulation of the 7-node network are shown in Fig. 2; note the convergence metric's floor value of $1.49 \cdot 10^{-9}$. The disruptive behaviours and the nodes' initial conditions are the same for both the ODDI-C and the Mean-based Algorithm cases. As seen in Fig. 2 (a), with ODDI-C implemented, consensus is rapidly and effectively reached—unlike the case with the mean-based algorithm (c). This is expressed more clearly looking at the convergence metric, Fig. 2 (d), which decreases for ODDI-C's run as compliant nodes come together and the difference between their state value reduces. This is not true for the mean-based algorithm simulation for which the convergence metric fluctuates along with the disruptive nodes' values.

The results of the simulation of the 15-node network are shown in Fig. 3. Here again, the difference between ODDI-C and the Mean-based Protocol runs are clear—with ODDI-C maximum convergence (floor value of the convergence metric) is reached in around 15 steps, while with the mean-based protocol, the state values of the compliant nodes wobble and proper consensus is not reached.

Table 1 Experiment 1 disruptive node parameters

	A	s_x	s_y	ω	m
$D1_{EXP1:7}$(T1)	0.4523	1.5011	0.42	1.8711	–
$D2_{EXP1:7}$(T1)	0.4512	3.2642	− 1.3117	0.2761	–
$D1_{EXP1:15}$(T2)	–	–	–	–	− 0.1707

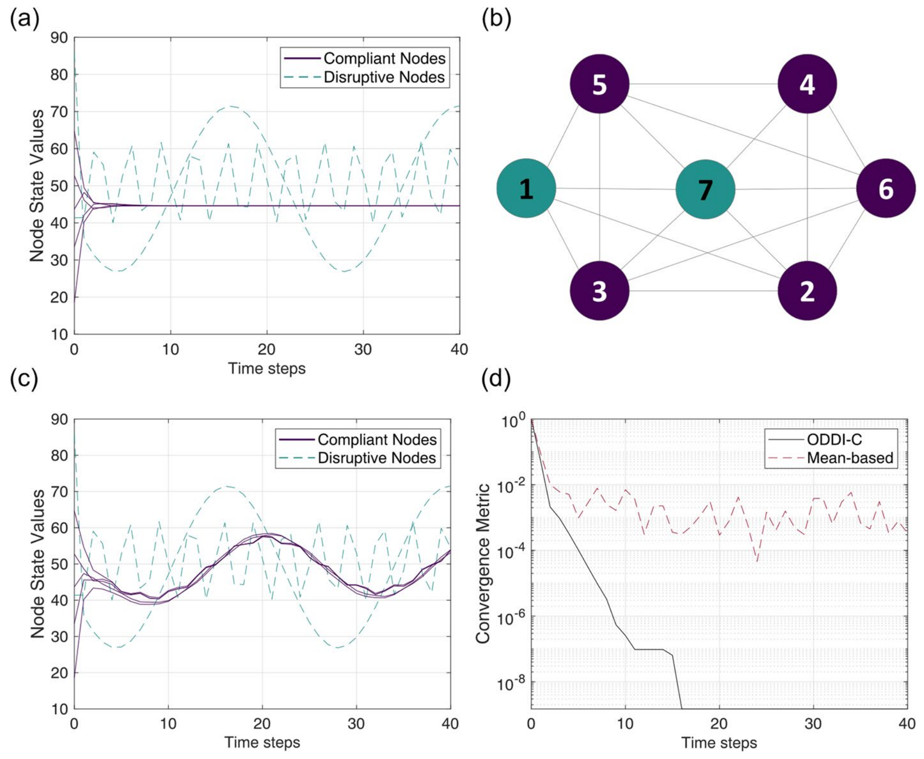

Fig. 2 Comparison of the ODDI-C and the Simple Mean-based Algorithm performances for the 7- Node (3,3)-robust Network with 2 Disruptive Agents. The two disruptive agents both exhibit a sine wave (T1, as described in Eq. 16), each with a different amplitude and period. Tiles (**a** and **c**) show the nodes' Opinion Trajectories for the ODDI-C (**a**) and the Mean-based algorithms (**c**), tile (**b**) shows the network diagram (compliant agents shown in purple) and tile (**d**) compares the scenarios' Convergence Metric (calculated as the normalised difference between compliant nodes, see Eq. 12)

Experiment 2: fixed number of disruptors, increasing network connectivity

A twenty-node network with a fixed number of disruptive nodes, $|D| = 5$, is considered. The ODDI-C protocol's performance is assessed when increasing the network connectivity (in-degree) in steps of two: $V^{in} = [3, 5, \ldots, 15]$. To account for the random nature of all consensus protocols, and of possible behaviours, Monte Carlo simulations of 50 runs per in-degree are performed. The results of the runs, meaning their convergence metrics, are averaged to showcase the trends of the simulations with error bars provided to show maximum and minimum values of each batch's convergence metrics. These are also used for the ODDI-C Algorithm's comparison with the MSR.

To allow for comparison, the behaviour of the disruptions is kept constant across runs and Monte Carlo batches ($50 \times 8 = 400$ simulations). The opinion trajectories are calculated for the first simulation, stored, and then re-used for subsequent simulations. Note however that while the behaviours are fixed, the identities of the nodes exhibiting this disruptive behaviour are not, being selected at random from the pool of all node for each run. The disruptors are three sine waves (T1, described in Eq. 16), a linear function (T2, described in Eq. 17), and a noise function (T3, described in Eq. 18). The parameters describing the disruptive behaviours are shown in Table 2.

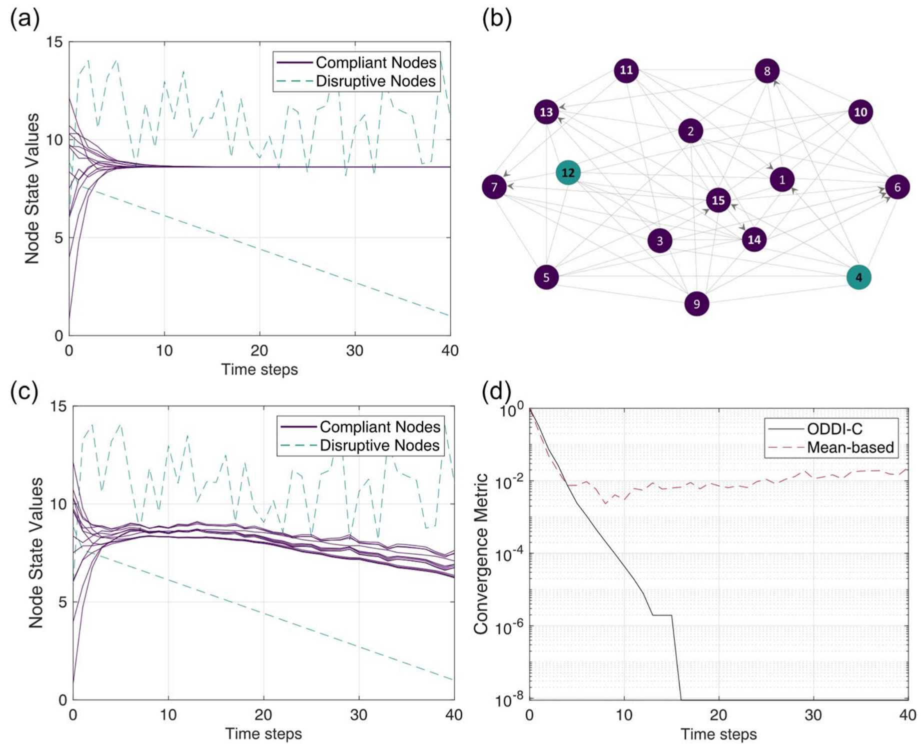

Fig. 3 Comparison of the ODDI-C and the Simple Mean-based Protocol performances for the 15-Node (3,2)-robust Network with 2 Disruptive Agents. The two disruptive agents exhibit are a noise pattern (T3, as described in Eq. 18) and a linear behaviour (T2, as described in Eq. 17) respectively. Tiles (**a** and **c**) show the nodes' Opinion Trajectories for the ODDI-C (**a**) and the Mean-based protocols (**c**) respectively, tile (**b**) shows the network diagram (compliant agents shown in purple) and tile (**d**) compares the scenarios' Convergence Metric (calculated as the normalised difference between compliant nodes, see Eq. 12)

Table 2 Experiment 2 disruptive node parameters

	A	s_x	s_y	ω	m
$D1_{EXP2}$(T1)	0.1137	1.2050	− 3.9058	3.1381	–
$D2_{EXP2}$(T1)	0.5181	0.9961	− 1.5561	0.2217	–
$D3_{EXP2}$(T2)	–	–	–	–	0.5738
$D4_{EXP2}$(T1)	0.39528	4.5602	− 0.0241	0.2203	–

For each run, a new uniformly random graph is constructed. Taking as input the batch's in-degree, each node's in-neighbours are selected uniformly at random, without replacement, from the vector of all other nodes in the system. This simple random selection of in-degrees allows for a maximum variation between the tested digraph networks and their resulting adjacency matrix.

The ODDI-C Algorithm's outperforms the MSR (a pattern repeated for other disruptive behaviours) as observed across Fig. 4. For each connectivity, ODDI-C reaches consensus with equal or better performance than MSR. When simulating high in-degrees, ODDI-C reaches consensus with equal or fewer time steps than MSR despite having no knowledge of the number of disruptive agents within the system. Lower in-degree

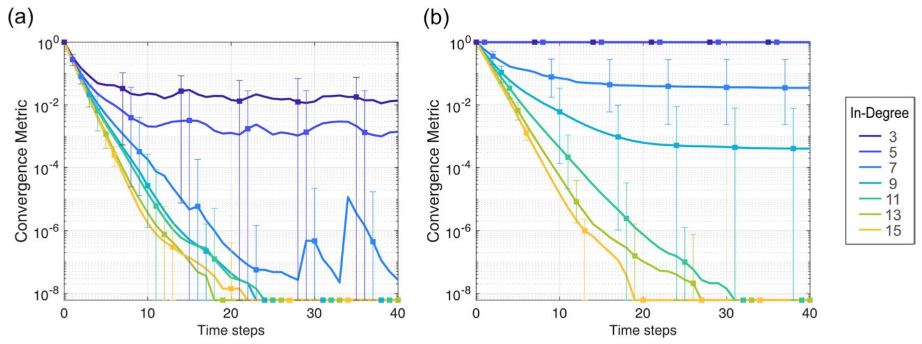

Fig. 4 Comparison of the (**a**) ODDI-C and (**b**) MSR Algorithms' convergence metrics

exhibit similar behaviour, due to the high ratio of disruptor nodes to connectivity, the MSR filters out most if not all values received by nodes. As a result, compliant nodes cannot reach any form of consensus (see tile (b) of Fig. 4). This is not the case of ODDI-C, which even with low in-degrees manages to bring nodes to some form of low-level consensus in the network.

Experiment 3: fixed network connectivity, increasing number of disruptors

A twenty node network with a fixed connectivity (in-degree of 6 for all agents), with an increasing number of unknown disruptors, $|D|$, starting with no disruptors and increasing in steps of 1 until reaching a maximum of 8 is considered: ($|D| = [0, 1, \ldots, 8]$). Monte Carlo simulations of 50 runs are again performed for each batch of disruptors. The results of the runs, meaning their convergence metrics, are averaged to showcase the trends of the simulations, with error bars showing maximum and minimum convergence metric values being shown at constant intervals. These are also used in the comparison with the MSR.

As with Experiment 2, to enable comparison between the ODDI-C and MSR algorithms, the behaviour of the disruptions is kept constant across runs, and Monte Carlo batches. For the first run of the second simulation batch (number of Disruptive agents ($|D| = 1$)), the opinion trajectory of the disruptive node is calculated before being stored. This opinion at each time step is then re-used for the other 49 runs of the batch ($|D| = 1$) as well as being used by later batches with more disruptive agents. Taking the case of $|D| = 7$, the behaviour of $D = 1$ is the same as for $|D| = 2, 3, \ldots, 8$. This continuity enables for a fairer comparison of not only runs, but also MSR against ODDI-C Algorithm. Note, again as with Experiment 2, that the node identity of the disruptors is non-constant, being selected at random from the pool of all node for each run. This Experiment's disruptions are four sine waves (T1, described in Eq. 16), a linear function (T2, described in Eq. 17), and three noise functions (T3, described in Eq. 18). The parameters of these disruptions are shown in Table 3.

The results of Experiment 3 are shown in Fig. 5. The convergence metrics are plotted for all investigated disruptor numbers, with error bars. As before, ODDI-C sees a performance equal or better than that of the MSR in all cases. For cases with few (0–3) disruptive agents, the convergence patterns are the similar for both ODDI-C and MSR. MSR reaches maximum consensus in around 25 s while ODDI-C reaches consensus in

Table 3 Experiment 3 disruptive node parameters

	A	s_x	s_y	ω	m
$D1_{EXP3}(T1)$	0.3666	1.6825	0.2100	1.2452	–
$D2_{EXP3}(T1)$	0.08841	4.3958	1.1349	1.12364	–
$D4_{EXP3}(T1)$	0.4889	0.4714	−0.0241	4.4269	–
$D7_{EXP3}(T1)$	0.6072	3.0984	−2.0524	2.4376	–
$D8_{EXP3}(T2)$	–	–	–	–	−0.3156

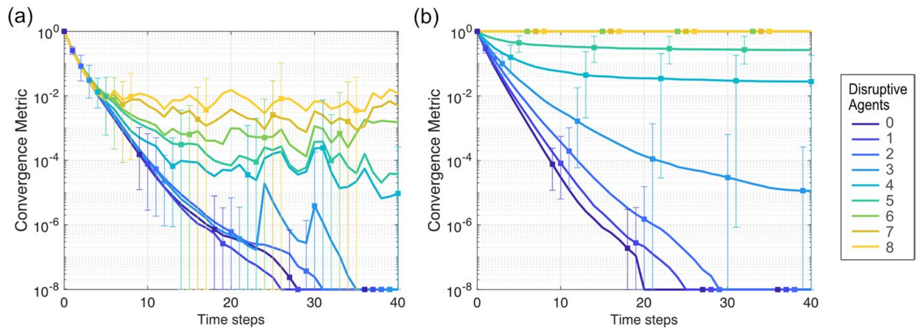

Fig. 5 Comparing the convergence metrics of the ODDI-C (**a**) and MSR (**b**) algorithms

almost 30 s. This trend reverses for higher numbers of disruptors. In these cases, ODDI-C's caution is rewarded as the disruption is observed to have less impact than on MSR's convergence.

Discussion

Numerical analysis of performance confirms that the ODDI-C Algorithm satisfies its design requirements, meaning the Termination, Agreement, and Weak Validity properties, whilst outperforming MSR when looking at simulation convergence metrics. Experiment 1, through its simulation of the 7-node and 15-node networks (Fig. 2a and Fig. 3a respectively), validates the ODDI-C algorithm's Weak Validity property, meaning that if all nodes are compliant and propose the same value, all nodes will agree on this value. This is further corroborated by Experiment 3's 0-Disruptor test which sees rapid consensus among all nodes. Experiments 2 and 3 validate ODDI-C's Termination and Agreement Properties with all compliant agents deciding on the same value (within tolerance).

Two key requirements of MSR-type methods are knowledge of the network topology and of the number of disruptors in the systems. These requirements constraint MSR's applicability to systems where such knowledge is available and fixed, meaning that the network properties or number of disruptors will not change. In contrast to these rigid implementation requirements, ODDI-C offers the clear benefit of adaptability in addition to a better performance than MSR-type methods, as demonstrated across all three experiments. In Experiment 1, ODDI-C's adaptability to various types of disruptive behaviours is shown with the algorithm minimising the influence of disruptors. Compared to the mean-based algorithm case, ODDI-C is observed to effectively dampen disruptive instabilities while ensuring convergence of compliant nodes. Similarly, Experiments 2 and 3 display ODDI-C's adaptability and its advantages compared to MSR, other than the already significant ability to operate without knowledge of the number of disruptive nodes in the network. In both experiments, MSR's trends show a near-constant convergence of compliant nodes until convergence plateaus are reached (see both Fig. 4b and Fig. 5b). This behaviour is expected from MSR's fixed filtering method. If the number of disruptors is known, filtering out twice that number of the most extreme values received will result in a steady convergence, unaffected by disruptive patterns. This is not the case for ODDI-C. The algorithm filters a non-constant number of values, each time adapting to the values received. As a result, filtering and convergence vary for each node across time steps and runs, as noted by ODDI-C's less constant rate of convergence in both experiments.

In experiment 2, the adaptability is illustrated when assessing both MSR's and ODDI-C's behaviours at high in-degrees. For MSR, increasing connectivity significantly improves the gradient of convergence, in a steady and consistent manner (see Fig. 4b). MSR's performance nonetheless remains limited by the amount of filtering being performed. Experiment 2 deals with 5 fixed disruptors, and thus up to 10 values will be filtered out (the 5 highest and lowest values received). In most cases, fewer values will be removed as only values strictly more extreme than the node's own values are removed. Despite this precaution, a vast number of non-malicious nodes will be filtered out, slowing consensus. This is not the case for ODDI-C whose filtering is dynamic and for which increasing connectivity drastically improves performance (see Fig. 4a). This improvement in convergence is the result of nodes being provided with more data points/samples from which to create their data distributions and from which to calculate z-scores for filtering (see Section: Algorithm Rules for details of the protocol's filtering). Increasing connectivity refines the filter, resulting in a faster convergence. While ODDI-C displays high convergence trends, especially when boosted by high in-degrees, it is worth acknowledging a by-product of this adaptability: when exhibiting non-extreme values, disruptive nodes may not be caught by the dynamic filtering and may thus impact the final convergence value. This by-product is observed when looking at ODDI-C's trends for relatively high in-degrees ($V^{in} = 7$) for which oscillations or "wobbles" are observed starting from time step 25. This behaviour, which stems from disrupting sine wave oscillations, emerges only in the ODDI-C simulations. Note, MSR's faces the same disruptors but is also less converged. This variation is the results of distributed consensus' dynamism, specifically, the fact that compliant nodes are always accounting for other nodes' values when creating their filters. Thus, in cases where disruptor nodes' values come close to connected nodes' medians, their disrupting values will influence compliant node's update, making them waver. This characteristic, the wobbling of compliant nodes' convergence, nonetheless remains small (see the log-scale used for the convergence metric).

ODDI-C's adaptability is similarly observed in Experiment 3's performance difference between MSR and ODDI-C. Specifically, for low numbers of disruptors. When dealing with no or low numbers of disruptions, the MSR acts like a classical mean-based consensus, removing none (or very few) of its neighbours. ODDI-C meanwhile, still filters in-coming values. As such, nodes will be accepting a reduced number of state values compared to the MSR, slowing down convergence (see Fig. 5). Despite this slight drawback, ODDI-C will still reach convergence; this vigilance is proven to be an advantage whenever disruptions are involved in any scenario. Against low numbers of disruptions, the algorithm's dynamism avoids over-filtering. Unlike the MSR, ODDI-C is therefore able to reach very high levels of convergence without plateauing when presented with disruptions (similar to the patterns observed in Fig. 4).

Finally, ODDI-C's adaptability is what allows it to provide a level of convergence when dealing with high ratios of disruptors to connectivity (batches with low connectivity in Fig. 4 and high number of disruptors in Fig. 5). The higher this ratio, the poorer the convergence, although this remains better than MSR's complete lack of consensus. On MSR's part, this is the result of filtering out all values. For ODDI-C, this stems from the calculation of a median-based z-score. That is, the fewer in-neighbours a node has, the less data points will it have to form a distribution of received values. Z-scores calculated will be coarser and less accurate, with the node's own z-score (which acts as filter discriminant) being more easily influenced by extreme nodes. High ratios of disruptor to connectivity are also much more likely to see one node being overwhelmed by disruptive nodes, and in turn acting disruptive to its own out-neighbours. Nevertheless, it is interesting to note that high levels of performance can still be reached for specific networks (see error bars) even in cases of low connectivities.

In summary, ODDI-C outperforms MSR thanks to its adaptability, despite lacking knowledge of the topology or number of disruptive agents. With its dynamic filtering approach, ODDI-C effectively filters disruptions for a range of disruptor number to connectivity ratios, demonstrating high levels of disruption-resilience. ODDI-C makes the most of increased connectivity and information when its nodes receive it, more so than MSR, as well as continuing to function in runs facing high disruptor ratios which overwhelm MSR. ODDI-C's social dynamics derived resilience is noted across all three experiments, confirming the algorithm as disruption-tolerant and validating its consensus properties across a range of scenarios.

Conclusion

The mechanics of social opinion dynamics can be used to engineer efficient disruption-tolerant consensus algorithms in fixed multi-agent networks requiring no knowledge of either network topology or the number of disruptive agents. By adapting the mechanics of tolerance-based opinion diffusion, the Opinion Dynamics-inspired Disruption-tolerant Consensus (ODDI-C) algorithm effectively dampens the instabilities generated by disruptor agents shown to more severely affect mean-based algorithm scenarios. By curbing the impact of disruptors, the ODDI-C algorithm enables high levels of convergence to be reached and confirms itself to be disruption-tolerance. The algorithm presented outperforms general and adapted MSR Methods, reaching a better convergence values for the same connectivity in addition to being more robust to the influence of disruptor nodes. The ODDI-C algorithm enables for better consensus in spite of the presence of disruptive agents and with less knowledge than that required by MSR.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1007/s41109-024-00671-x.

Additional file 1

Acknowledgements
Not applicable.

Author contributions
Conceptualization and methodology, A.B., R.C., C.L., and M.M.; software and numerical experiments, A.B.; Interpretation, A.B., R.C., and M.M.; writing, review, and editing, A.B., R.C., and M.M.; funding acquisition, M.M, and R.C. All authors have read and agreed to the published version of the manuscript.

Funding
This project is funded by the Air Force Office of Scientific Research (AFOSR), Grant number FA8655-22-1-7033.

Availability of data and materials
ODDI-C algorithm is freely available on GitHub. https://doi.org/10.5281/zenodo.10650711.

Declarations

Competing interests
The authors declare no competing interests.

Received: 13 February 2024 Accepted: 16 September 2024
Published online: 30 September 2024

References

Abbas W, Laszka A, Koutsoukos X (2017) Improving network connectivity and robustness using trusted nodes with application to resilient consensus. IEEE Trans Control Netw Syst 5:2036–2048
Al-KuwaitiM, Kyriakopoulos N, Hussein S (2006) Network dependability, fault-tolerance, reliability, security, survivability: a framework for comparative analysis. In: 2006 International conference on computer engineering and systems. pp. 282–287
Araque O, Iglesias CA (2020) An approach for radicalization detection based on emotion signals and semantic similarity. IEEE Access 8:17877–17891
Azadmanesh MH, Kieckhafer RM (2000) Exploiting omissive faults in synchronous approximate agreement. IEEE Trans Comput 49:1031–1042
Berghel H (2018) Malice domestic: the Cambridge analytica dystopia. Computer 51:84–89
Castellano C, Fortunato S, Loreto V (2009) Statistical physics of social dynamics. Rev Mod Phys 81:591
CivitP, Gilbert S, Guerraoui R, Komatovic J, Vidigueira M (2023) On the validity of consensus. In: Proceedings of the 2023 ACM symposium on principles of distributed computing. pp. 332–343
CorettiS, Kiayias A, Moore C, Russell A (2022) The Generals' Scuttlebutt: byzantine-resilient gossip protocols. In: Proceedings of the 2022 ACM SIGSAC conference on computer and communications security. pp. 595–608
DeffuantG, Amblard F, Weisbuch G, Faure T (2002) How can extremism prevail? A study based on the relative agreement interaction model. J Artif Soc Social Simul. 5
DeffuantG, Jager W, Moss W (2006) Dialogues concerning a (possibly) new science. J Artif Soc Social Simul. 9
Ding D, Han Q-L, Xiang Y, Ge X, Zhang X-M (2018) A survey on security control and attack detection for industrial cyber-physical systems. Neurocomputing 275:1674–1683
Ferrari RM, Teixeira AM (2021) Safety, security and privacy for cyber-physical systems. Springer
Garay JA (1994) Reaching (and maintaining) agreement in the presence of mobile faults. In: International workshop on distributed algorithms. Springer. pp. 253–264
Guarino S, Trino N, Celestini A, Chessa A, Riotta G (2020) Characterizing networks of propaganda on twitter: a case study. Appl Netw Sci 5:1–22
Ishii H, Wang Y, Feng S (2022) An overview on multi-agent consensus under adversarial attacks. Ann Rev Control
Jadbabaie A, Lin J, Morse AS (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans Autom Control 48:988–1001
Kieckhafer RM, Azadmanesh MH (1994) Reaching approximate agreement with mixed-mode faults. IEEE Trans Parallel Distrib Syst 5:53–63
LeBlanc HJ, Zhang H, Koutsoukos X, Sundaram S (2013) Resilient asymptotic consensus in robust networks. IEEE J Sel Areas Commun 31:766–781
Leys C, Ley C, Klein O, Bernard P, Licata L (2013) Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. J Exp Soc Psychol 49:764–766
Lv Y, Wen G, Huang T (2019) Adaptive protocol design for distributed tracking with relative output information: a distributed fixed-time observer approach. IEEE Trans Control Netw Syst 7:118–128
Mathias J-D, Huet S, Deffuant G (2016) Bounded confidence model with fixed uncertainties and extremists: the opinions can keep fluctuating indefinitely. J Artif Soc Soc Simul 19:6

Michiardi P, Molva R (2002) Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Jerman-Blažič B, Klobučar T (eds) Advanced communications and multimedia security. Springer, pp 107–121. https://doi.org/10.1007/978-0-387-35612-9_9
Murphy KP (2012) Machine learning: a probabilistic perspective. MIT press
Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. Proc IEEE 95:215–233
Pasqualetti F, Bicchi A, Bullo F (2011) Consensus computation in unreliable networks: a system theoretic approach. IEEE Trans Autom Control 57:90–104
Phan L-A, Kim T (2020) Fast consensus-based time synchronization protocol using virtual topology for wireless sensor networks. IEEE Internet Things J 8:7485–7496
Pierri F, Artoni A, Ceri S (2020) Investigating Italian disinformation spreading on Twitter in the context of 2019 European elections. PLoS ONE 15:e0227821
RaynalM (2002) Consensus in synchronous systems: a concise guided tour. In: Pacific Rim international symposium on dependable computing. pp. 221-228
RenganathanV, Summers T (2017) Spoof resilient coordination for distributed multi-robot systems. In: 2017 International symposium on multi-robot and multi-agent systems. (MRS), IEEE. pp. 135–141
Rousseeuw PJ, Croux C (1993) Alternatives to the median absolute deviation. J Am Stat Assoc 88:1273–1283
Sakavalas D, Tseng L (2019) Network topology and fault-tolerant consensus. Synth Lect Distrib Comput Theory 9:1–151
SaldanaD, Prorok A, Sundaram S, Campos MF, Kumar V (2017) Resilient consensus for time-varying networks of dynamic agents. In: 2017 American control conference (ACC), IEEE. pp. 252–258
Senejohnny DM, Sundaram S, De Persis C, Tesi P (2019) Resilience against misbehaving nodes in asynchronous networks. Automatica 104:26–33
Shang Y (2021) Median-based resilient consensus over time-varying random networks. IEEE Trans Circuits Syst II Express Briefs 69:1203–1207
Sobkowicz P (2015) Extremism without extremists: Deffuant model with emotions. Front Phys 3:17
Su L, Vaidya NH (2017) Reaching approximate byzantine consensus with multi-hop communication. Inf Comput 255:352–368
TrejoJAV, Rotondo D, Medina MA, Theilliol D (2020) Observer-based event-triggered model reference control for multi-agent systems. In: 2020 International conference on unmanned aircraft systems (ICUAS). pp. 421–428
Wang Y, Ishii H, Bonnet F, Defago X (2021) Resilient real-valued consensus in spite of mobile malicious agents on directed graphs. IEEE Trans Parallel Distrib Syst 33:586–603
Weisbuch G, Deffuant G, Amblard F, Nadal JP (2002) Meet, discuss, and segregate! Complexity 7:55–63
Xie J, Sreenivasan S, Korniss G, Zhang W, Lim C, Szymanski BK (2011) Social consensus through the influence of committed minorities. Phys Rev E 84:011130
YuanL, Ishii H (2022) Asynchronous approximate Byzantine consensus via multi-hop communication. In: 2022 American control conference (ACC). 2022, pp. 755–760
Zhai Y, Liu Z-W, Ge M-F, Wen G, Yu X, Qin Y (2020) Trusted-region subsequence reduction for designing resilient consensus algorithms. IEEE Trans Netw Sci Eng 8:259–268
ZhangH, Sundaram S (2012) Robustness of information diffusion algorithms to locally bounded adversaries. In: 2012 American control conference (ACC). pp. 5855–5861

Publisher's Note