Effective Collective Summarisation of Distributed Data in Mobile Multi-Agent Systems

Giorgio Audrito
University of Torino
Torino, Italy
giorgio.audrito@unito.it

Sergio Bergamini
University of Torino
Torino, Italy
sergio.bergamini@edu.unito.it

Ferruccio Damiani
University of Torino
Torino, Italy
ferruccio.damiani@unito.it

Mirko Viroli
University of Bologna
Cesena, Italy
mirko.viroli@unibo.it

ABSTRACT

One of the key applications of physically-deployed multi-agent systems, such as mobile robots, drones, or personal agents in human mobility scenarios, is to promote a pervasive notion of distributed sensing achieved by strict agent cooperation. A quintessential operation of distributed sensing is data summarisation over a region of space, which finds many applications in variations of counting problems: counting items, measuring space, averaging environmental values, and so on. A typical strategy to perform peer-to-peer data summarisation with local interactions is to progressively accumulate information towards one or more collector agents, though this typically exhibits several sources of fragility, especially in scenarios featuring high mobility.

In this paper, we introduce a new multi-agent algorithm for dynamic summarisation of distributed data, called *parametric weighted multi-path*, based on a local strategy to break, send, and then recombine sensed data across neighbours based on their estimated distance, ultimately resulting in the formation of multiple, dynamic and emergent paths of information flow towards collectors. By empirical evaluation via simulation in synthetic and realistic case studies, accounting for various sources of volatility, using different state-of-the-art distance estimations, and comparing to other existing implementations of aggregation algorithms, we show that *parametric weighted multi-path* is able to retain adequate accuracy even in high-variability scenarios where all other algorithms are significantly diverging from correct estimations.

KEYWORDS

data aggregation; adaptive algorithm; aggregate programming; computational field; gradient

ACM Reference Format:
Giorgio Audrito, Sergio Bergamini, Ferruccio Damiani, and Mirko Viroli. 2019. Effective Collective Summarisation of Distributed Data in Mobile Multi-Agent Systems. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 9 pages.

1 INTRODUCTION

The modern world is increasingly permeated by heterogeneous connected, intelligent and mobile computing devices (smartphones, drones, robots). Such a landscape increasingly calls for the adoption of collective adaptation mechanisms so as to fulfil its potential of forming a true *pervasive computing fabric*, where sensing, actuation and computation can be naturally seen as inherently distributed across physical space [11].

Each computational device can be modelled as (or programmed with) an agent with limited knowledge of the portion of environment it operates upon, so the problem naturally turns into one of engineering a situated multi-agent system (MAS) so as to effectively, efficiently, and robustly turn pointwise knowledge items into single global knowledge. Most specifically, we seek for the adoption of self-organisation techniques [28, 34] to build MASs able to realise distributed sensing, concerning physical properties of the environment or virtual/digital characteristic of the computational one: it is by the strict cooperation and interaction of dynamic sets of mobile agents situated in proximity that it can be possible to support complex situation recognition [14], aggregation of agent intentions/predictions [13, 30], better monitoring of physical environment [11], and observation (and then control) of teams of agents [33]. Several framework and tools are proposed to address this kind of problems [7, 26], showing how self-organisation can be achieved using a small set of "basic" interaction laws, upon which increasingly complex behaviours are built [9, 16]. Due to the dynamic scenarios typical of pervasive computing, these laws need to carefully trade-off efficiency with resilience to network changes.

A quintessential operation of distributed sensing is data summarisation over a region of space, which finds applications in operations such as counting, integration, averaging, maximisation, density estimation, and so on. This problem can be solved by a *collection* distributed algorithm (sometimes named the "C" building block, in short [31]), which is one of the most basic and widely used components of collective adaptive systems (CASs): it aggregates values held by the mobile agents (/devices) situated in a spatial region into a single resulting value in a selected agent (/device) called *collector*. Seen as a distributed process, collection is essentially a multi-agent process that computes a specific case of "computational field" [20, 33], namely, what one can interpret as a *knowledge field*: knowledge distributed across space such that each agents perceives

only the local value of it—e.g., representing a partial result of counting for a sub-region. Distributed collection resembles the *reduce* phase of the MapReduce paradigm [15] ported into a "spatial" context of agents spread in a physical environment and communicating by proximity, and has close analogues designed for wireless sensor networks [29]. This "brick" can be applied to a variety of different contexts, as it can be instantiated for values of any data type with an associative and commutative aggregation operator.

However, it turns out that efficiently implementing C is tricky: existing implementations (single-path and multi-path [31]) are very fragile (inaccurate, slow, and non-resilient) to mobility of agents (and hence, to changes in the network of computational devices), which is the norm in several emerging application contexts, including airborne sensing by drones [10], crowd management by people smartphones [9], and vehicular networks [22].

In this paper we present a new algorithm for effectively and efficiently carrying on the computation of the C building block, called *parametric weighted multi-path* (C_{pwmp}), which is able to achieve adequate accuracy in highly volatile scenarios. As for existing multi-path collection algorithms, in C_{pwmp} data chunks flow through agents by every possible link of the underlying proximity network. Differently from them, however, data chunks are weighted and broken in pieces with carefully chosen factors reflecting the probability for the link to be lost in the immediate future because of mobility. Moreover, the algorithm is parametric in the form of aggregation used, allowing its usage both for *arithmetic* and *idempotent* aggregation operators. We validate its performance in archetypal situations of (respectively) *device counting* and *progress tracking*, taking into account agent mobility, update rate variability and discontinuities in network configuration. Finally, a realistic case study of crowd safety services on real GPS traces is carried out to further validate the approach. Ultimately, by accounting for various sources of volatility, using different state-of-the-art distance estimations, and comparing to other existing implementations of aggregation algorithms, we show that C_{pwmp} is able to retain acceptable precision even in high-variability scenarios where all other algorithms are significantly diverging from correct estimations.

The work of this paper is arguably a significant step in the context of engineering MASs and CASs. In general, the proposed algorithm can be used as a solid component for engineering collection services in highly distributed and mobile multi-agent systems, e.g., as an orthogonal distributed knowledge component for intelligent agent frameworks with a designed notion of environment and organisation [18, 27], or for agent planning as of [33]. On the other hand, in the specific context of the aggregate computing framework [9], C_{pwmp} provides an implementation for the fundamental "C block" as advocated in [31], coupling that of "G block" as of [4], and together forming a set of combinators effectively supporting construction of higher-level, self-stabilising computations in mobile distributed systems: these are available as parts of the library of Protelis [25], an implementation of the aggregate computing framework.

The remainder of this paper is organised as follows. Section 2 presents the background and related works, focussing on the state-of-the-art implementations of C. Section 3 describes the new C_{pwmp} algorithm. Section 4 compares the new algorithm with other existing implementations in archetypal scenarios and in a case study. Section 5 summarises the contributions of this paper and outlines possible future works for further improving the C_{pwmp} collection block.

2 BACKGROUND AND RELATED WORK

The C_{pwmp} algorithm generalizes recently proposed techniques for summarizing distributed data [3] by providing support for aggregations through different kinds of operations (including idempotent operators), and by introducing the *flow control* technique for reducing error peaks during input discontinuities.

Data aggregation (also called *collection*) is a crucial component of distributed algorithms. As such, it has been tacked in different ways depending on the application context: high-performance computing [15], wireless sensor networks [23, 29] and spatial computing [8]. However, all of these different approaches rely on the same basic mechanisms. In data aggregation, distributed values have to be combined together through an aggregation operator $\oplus$ satisfying the following properties:

(1) *commutativity*: $u \oplus v = v \oplus u$;
(2) *associativity*: $u \oplus (v \oplus w) = (u \oplus v) \oplus w$.

Provided that these two axioms hold, the aggregation $\bigoplus C$ of a multi-set C is well-defined, regardless of the overall order in which the individual elements are aggregated. Among others, common aggregation operators are: addition, multiplication, maximum and minimum. In scenarios with intrinsic communication errors and input volatility (such as wireless sensor networks and spatial computing), a further more informal requirement has to be considered:

(3) *continuity*: the effect on the aggregation of a certain percentage p of errors tends to zero as p tends to zero.

This property holds for the aggregation operators cited above, however, it does not hold for other operations such as modular sum: the modular addition of a single spurious element is enough to fully disrupt the outcome of the aggregation of a massive collection of elements.

Given a commutative and associative operator, and an environment with proximity-based interactions, a data aggregation routine thus asynchronously combines values held by different devices into a single value in a selected device (called *source*, or sometimes *collector*), controlling the flow of data towards the source in order to avoid multiple aggregation of the same values. This two-faceted prerequisite, of *acyclic* flows *directed* towards the source, is met by relying on a given *potential field*, approximating a certain measure of distance from the selected source. As long as information flows descending the potential field, cyclic dependencies are prevented and eventual reaching of the source is guaranteed. Potential descent is enforced by splitting the neighbours D_δ (i.e., devices able to directly communicate with δ) of a device δ according to their potential value $P(\cdot)$, obtaining the two disjoint sets:

$$D_\delta^- = \{\delta' \text{ linked to } \delta \text{ such that } P(\delta') < P(\delta)\}$$
$$D_\delta^+ = \{\delta' \text{ linked to } \delta \text{ such that } P(\delta') > P(\delta)\}$$

Two main implementation strategies of the collection block have been proposed so far: *single-path* and *multi-path*, both scaling to arbitrarily large systems (as they require constant computational resources per node). Furthermore, they can be integrated with

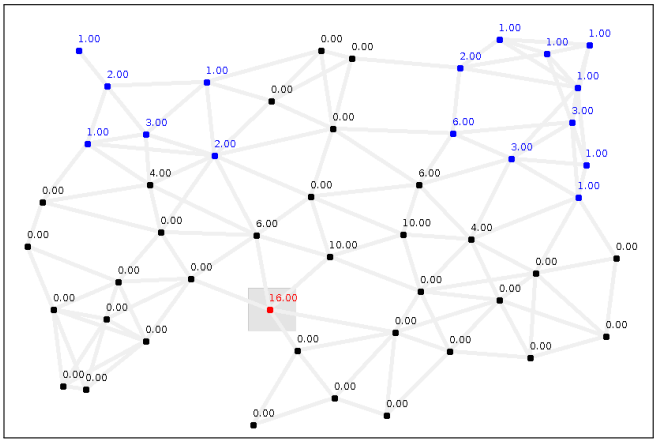

Figure 1: An example of collection field in a p2p scenario, using single-path aggregation, counting the number of blue agents, and collecting the result in the red agent; note that each agent holds a partial result of counting, based on how many "single-path flows" from blue agents to red agent cross it.

standard trust techniques to provide security against malicious data sources.

2.1 Single-path Aggregation

The single-path strategy C_{sp} ensures that information flows through a forest in the network (so that no multiple aggregation is possible), by sending the whole partial aggregate of a device δ to the single device $m(\delta)$ with *minimal* potential among devices connected to δ. This is accomplished by repeatedly applying the following rule:

$$C_{\text{sp}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D^+_\delta \wedge m(\delta')=\delta} C_{\text{sp}}(\delta'),$$

which computes the partial aggregate in δ by combining together the value v_δ held in δ and the partial aggregates in devices with higher potential for which δ is the selected output device $m(\delta')$.

Since data flows descending the potential as fast as possible, single-path aggregation attains optimal reactivity to input changes in static environments. However, in mutable environments, $m(\delta)$ may not be able to receive the message from δ, disrupting communication and pruning the entire branch of the forest rooted in δ. This phenomenon translates into poor performances, provided that values far from the source contribute significantly to the aggregation (e.g., non-zero values for summation, high values for minimisation, and so on). Seen as a field computation, single-path aggregation results in a configuration of values as shown in Figure 1. As the details of the shape of such field can vary from implementation to implementation, the key aspect is that on sources of values to be collected, the field gives the value itself, while on the target agent the field gives the overall result of collecting.

2.2 Multi-path Aggregation

The multi-path strategy C_{mp} allows information to flow through every path compatible with the given potential field. In order to avoid double counting, it is thus necessary to divide the partial aggregate of a device δ equally among *every* device δ' connected to δ with lower potential, by iteratively applying the following rule:

$$C_{\text{mp}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D^+_\delta} \left\{ C_{\text{mp}}(\delta') \oslash |D^-_{\delta'}| \right\};$$

where $\oslash$ is a binary operator such that $v \oslash n$ means "dividing by n", i. e., an element that aggregated with itself n times produces the original value v. Since information needs to be "divisible" for $\oslash$ to exist, two categories of aggregation operators are supported:

(1) *arithmetic operations*, e.g., point-wise sum and multiplication of vectors $\vec{v} \in \mathbb{R}^n$ of real numbers (for which $\oslash$ is respectively division and root extraction);
(2) *idempotent operations*, e.g., computation of maximum and minimum among values v in a partially ordered set (for which $\oslash$ is the identity function).

Thus, theoretically, multi-path has a narrower scope than single-path. However, the vast majority of practically occurring (continuous) aggregation operators can be typically recast to be either arithmetic or idempotent: e.g., idempotent operations have been used to emulate several different aggregations through statistical tools: distinct count, sum, uniform sampling, selection of most frequent values [23], and order statistics [36].

Since data flows through every possible path, it is unlikely for devices to be excluded from the aggregation, making C_{mp} resilient to device mobility. On the other hand, the reactivity to input changes of multi-path aggregation is particularly poor. In fact, even in static environments, values flow through every possible path including the *longest* path, forcing reaction to changes to be delayed until all paths have been exploited (in particular for idempotent operations), and resulting in a reaction speed inversely proportional to the device density. In mutable environments, the problem is further exacerbated by the creation of *information loops*, which occur when two or more moving devices of similar potential invert their relative potential order in consecutive rounds, causing information from a device δ to come back to the same device, slowing down even further the reaction speed of the algorithm.

2.3 Distance Estimation Algorithms

All the above mentioned procedures for aggregating distributed data are based on a given *potential field* as input to guide the aggregation, representing a notion of physical distance from neighbour sources. Accurately computing distances in a distributed and volatile scenario is a demanding task, which can be tackled in different ways depending on the context. In spite of variations, the general framework is that of gradient-based *field computations* [21, 31], where local estimates from the source are repetitively shared with neighbours and combined with proximity estimates of mutual distance.

If no proximity sensors are available, the harsh *hop-count* measure can be improved through statistical tools [19], obtaining continuous and adaptive distance estimates. Furthermore, even when a proximity sensor is available, reactivity to input changes and network variability may be impaired by the *rising value problem*[1]—simply, reaction to changes causing increase of distance is very low [5]. Several solutions have been proposed to tackle this problem.

[1] Also known as the *count to infinity* problem in routing algorithms.

Following recent reviews of distance estimation algorithms [4, 5] three solutions are shown to always outperform basic algorithms: FLEX [6], BIS [5], and ULT [4].

FLEX is an algorithm aimed at maximising stability of values while containing the error within predictable bounds, which also addresses the rising value problem by introducing a metric distortion. BIS, instead, exploits time information in order to solve the rising value problem obtaining optimal single-path reactivity to input changes, without concerns on value stability. ULT develops on BIS by adding a stale values detector running at (faster) multi-path speed, while addressing value stability with the addition of filters and dampers. Being obtained by the integration of different methods, ULT is tuned by a large number of parameters, and can range to being almost identical to BIS (when filters and dampers are disabled) to being closer to FLEX (when dampers are active). Since we already included BIS in our sample of distance estimation algorithms, we chose ULT parameters according to the "stabilised" version evaluated in [4].

3 PARAMETRIC WEIGHTED MULTI-PATH

The C_{pwmp} collection develops on the multi-path strategy for arithmetic operations, by allowing partial aggregates to be divided unequally among neighbours. *Weights* corresponding to neighbours are calculated in order to penalise devices that are likely to lose their "receiving" status, a situation that can happen in two cases:

(1) if the "receiving" device is too close to the edge of proximity of the "sending" device, it might step outside of it in the immediate future breaking the connection;
(2) if the potential of the "receiving" device is too close to the potential of the "sending" device, their relative role of sender/receiver might be switched in the immediate future, possibly creating an "information loop" between the two devices.

We can address both of these situations with a natural weight function $w(\delta, \delta') = d(\delta, \delta') \cdot p(\delta, \delta')$, measuring how much of the information from δ should flow to δ' as the product of the two corresponding factors $d(\delta, \delta') = R - D(\delta, \delta')$ and $p(\delta, \delta') = |P(\delta) - P(\delta')|$, where R is the communication radius, $D(\delta, \delta')$ is the physical distance between devices δ, δ' and $P(\delta)$ is the potential of device δ. Notice that w is positive and symmetric, hence it can be interpreted as an attribute of connection links representing both the amount of information to "send" and to "receive".

Since these weights do not sum up to any particular value, they need to be normalised by the factor $N(\delta) = \sum_{\delta' \in D^-_\delta} w(\delta, \delta')$, obtaining normalised weights $w(\delta, \delta')/N(\delta')$. The partial aggregates accumulated by devices can then be calculated as in C_{mp} with the addition of weights, by iteratively applying the following rule:

$$C_{\text{pwmp}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D^+_\delta} \left\{ C_{\text{pwmp}}(\delta') \otimes \frac{w(\delta', \delta)}{N(\delta')} \right\};$$

where $\otimes$ is a binary operator such that $v \otimes k$ "extracts" a certain percentage k of a local value v. In particular, if $\oplus$ is addition then $\otimes$ is multiplication, whereas if $\oplus$ is multiplication then $\otimes$ is exponentiation. The operations $\oplus, \otimes$ are called *parameters* of the algorithm.

3.1 Implementation as Field Computation

For the sake of reproducibility, we report an implementation for C_{pwmp} as a field computation, expressed in `Protelis` language [9, 25], an incarnation of the field calculus [31], which is at the basis of the experiments described in next section. `Protelis` is a fully-functional language, where functions express transformation of whole, system-wise data structures (fields), but can also be interpreted as local declarative computations for the single device. Ad-hoc constructs to express field mechanisms include `rep`, used to evolve a field over time by iteratively applying the function expressed in its body, `nbr`, used to observe the result of evaluating its argument expression in neighbours, and `foldHood/sumHood` used to collapse such multiple observations into single values—see [25] for more details.

```
def weight(potential, radius) {
  (radius - nbrRange()) * (nbr(potential) - potential)
}
def normalize(w) {
  let sendTo  = max(-w, 0);
  let N = nbr(sumHood(sendTo));
  mux (N != 0) { max(w, 0) / N } else { 0 }
}

def Cpwmp(potential, radius, v, null, aggregate, extract) {
  rep (x <- v) {
    let nbrVals = extract(nbr(x),normalize(weight(potential,radius)));
    let nbrAggr = foldHood(aggregate, nbrVals, null);
    aggregate(v, nbrAggr)
} }
```

The `weight` function implements $w(\delta, \delta')$, using `nbrRange()` to compute the field of distances from neighbours and `nbr(potential)` to obtain the field of neighbours' potential values. The `normalize` function divides the incoming weights by the sum of outgoing weights for their respective neighbour `nbr(sumHood(sendTo))`. The main function `Cpwmp` uses the `extract` function to apply the normalized weights to the neighbours' values, and then aggregates all of them together with the local value `v`. The provided implementation can be used to transparently sustain field computation in each agent by the `Protelis` platform, such that each agent can perceive the local results of collection like with any other physical/virtual sensor.

3.2 Parametrisation

The above algorithm is naturally understood for parameters $\oplus = +$, $\otimes = \times$ (or in Protelis, `(a,b) -> {a+b}` and `(v,k) -> {v*k}`). However, it can apply to multiple different situations, including to *idempotent* aggregation (e.g., maximum or minimum). Since the "root extraction" corresponding to idempotent operations is the identity, weights cannot continuously influence the transmission of values. However, they can regulate which links should be exploited for transmission and which should be ignored through a threshold θ carefully determined based on the density of devices, since a higher number of neighbours results in lower normalised weights. In order to normalise the choice of θ, we thus ignore a link from δ' to δ if its normalised weight $w(\delta, \delta')/N(\delta')$ is below $\theta/|D_{\delta'}|$.

This procedure can thus be modelled by the same algorithm C_{pwmp} previously defined, by setting the parameter $\otimes$ equal to:

$$v \otimes k = \begin{cases} v & \text{if } k \geq \frac{\theta}{|D_{\delta'}|} \\ 0_\oplus & \text{otherwise} \end{cases}$$

where $0_{\oplus}$ is the null element of the $\oplus$ operation and θ is the threshold (in Protelis, `(v,k) -> {if (k >= t/countHood()) {v} else {null}}`). Notice that we are mathematically guaranteed that at least one neighbour device has weight k above $1/|D_{\delta'}|$, thus thresholds $\theta \leq 1$ force links to not be fully discarded. In practice, higher values can also be safely used provided that the network density is high enough, as we shall show in Section 4.2.

3.3 Flow Control

During an aggregation process, distributed information flows towards a set of (one or multiple) sources. In this context, the process generates a dynamic partition S of the whole set of devices into regions, where the cardinality of S equals the number of source devices disseminated in the network, and each element of S consists of exactly one source δ together with those devices for which δ is the nearest aggregation point. In fact, each element of the partition identifies a separate flow of information: devices in different flows should not interact as their values are directed towards different sources.

However, the collection algorithms presented so far do *not* strictly follow this "best practice": neighbour devices in the border of different regions *can* communicate, since one of them has to hold a higher potential than the other. Thus, the aggregate computed by a source is poisoned by values belonging to other flows of information. In static environments, the poisoning is restrained to border devices, resulting in a moderate to small overall error. In mutable environments, more significant errors can happen as devices cross the border of different regions: when the set of sources changes, values computed in a certain direction can flow back towards the opposite direction as they enter a new flow, causing extreme double-counting phenomena and exponential error growth.

Interaction between different flows can be avoided by labelling each flow with a unique identifier $f(\delta)$: e.g., a random number generated by the source of the flow and carried over by the distance calculation algorithm. Then, the parametric weighted multi-path collection can be improved as follows:

$$C_{\text{pwmpf}}(\delta) = v_\delta \oplus \bigoplus_{\delta' \in D^+_\delta \wedge f(\delta)=f(\delta')} \left\{ C_{\text{pwmpf}}(\delta') \otimes \frac{w(\delta', \delta)}{N(\delta')} \right\}.$$

4 EVALUATION

We compared C_{pwmp} with and without flow control (*pwmpf*, *pwmp*) against reference multi-path and single-path (*mp*, *sp*) implementations, both in isolation tests and in a realistic case study. For each of them, we evaluated the performance with respect to the state-of-the-art distance algorithms FLEX [6], BIS [5] and ULT [4] (see Section 2.3).

For the isolation tests, the same archetypal scenario was selected according to the guidelines developed in [5]:[2] 1000 devices randomly distributed along a $2000m \times 200m$ rectangular area, with a $1s$ average computation rate and a $100m$ communication range [2]. A single source device was located on the right end of the corridor, then discontinuously moved to the left end at time $T = 250s$. We tested degrees of variability ranging from 0 (no movement, regular computation rate for every device) to 1 (short- and long-range random movements with similar speed for every device, irregular computation rate in each device and between different devices), thus testing the algorithm in scenarios with increasing variability aimed at reproducing the worst possible case. The simulations were obtained with Protelis [25] as programming language, Alchemist as simulator [24] and the supercomputer OCCAM [1] as platform.[3]

4.1 Isolation Test: Device Counting

Firstly, we tested collection for arithmetic operators by setting $\oplus = +$, $\otimes = \times$, $\oslash = /$ and values $v_\delta = 1$ for each device. This choice amounts to *counting the total number of devices*, which is a commonly used routine and a paradigmatic example of arithmetic aggregation. We run 20 instances of each scenario and computed median results, as the relative standard errors between runs were significantly high. Figure 2 summarises the evaluation results.

Single-path collection systematically underestimates the ideal value, independently from the choice of the distance estimation algorithm, with a similarly poor performance under variabilities from 0.5 to 1 showing that accurate values are attainable only for very low-variability scenarios. Conversely, multi-path collection systematically overestimates the value with an exponentially-growing behaviour (randomly "reset" from time to time) which gets exponentially worse as variability increases.

On the other hand, C_{pwmp} collection with BIS distance estimation achieves an adequate accuracy even in highly volatile scenarios, scoring the best results for every value of variability, only slightly improved during input discontinuities by the addition of flow control (Section 3.3). In combination with FLEX or ULT, the performance decreases (while still being competitive with those of multi- and single-path collection) showing a preference for responsive estimates over stable ones.

We remark that the difference with the other algorithms is so significant that the proposed algorithm can be considered as *effectively extending the applicability of distributed collection in scenarios where it was previously inapplicable because of high degrees of variability.*

4.2 Isolation Test: Progress Tracking

We tested collection for idempotent operators by setting $\oplus = \max$ and threshold $\theta = 3.5$, which was set according to simulations (Figure 3) opting for the highest value granting negligible transmission failure (for the given density $\rho = 25$). The values to be aggregated were chosen to make the aggregation as difficult as possible. As discussed in Section 2, a difficult aggregation problem requires both *obsolete* and *distant* values to be able to significantly contribute to the aggregation. If obsolete values have a negligible impact, multi-path collection is optimal as it does not need to react to environmental changes. If distant values have a negligible impact, single-path collection is optimal since even a small coverage of the network may be sufficient.

These two conditions hold true for any arithmetic aggregation of non-null values and may seem trivial, however, most simple idempotent aggregation problems fail to satisfy one of them. For

[2] The guidelines require high number of hops from the source to extremal nodes, and source discontinuities for measuring reaction to changes, in order provide a general and significant testing bed for this kind of algorithms.

[3] The URL of the actual code experiment is not indicated to preserve anonymity; it will be made available at publication time.

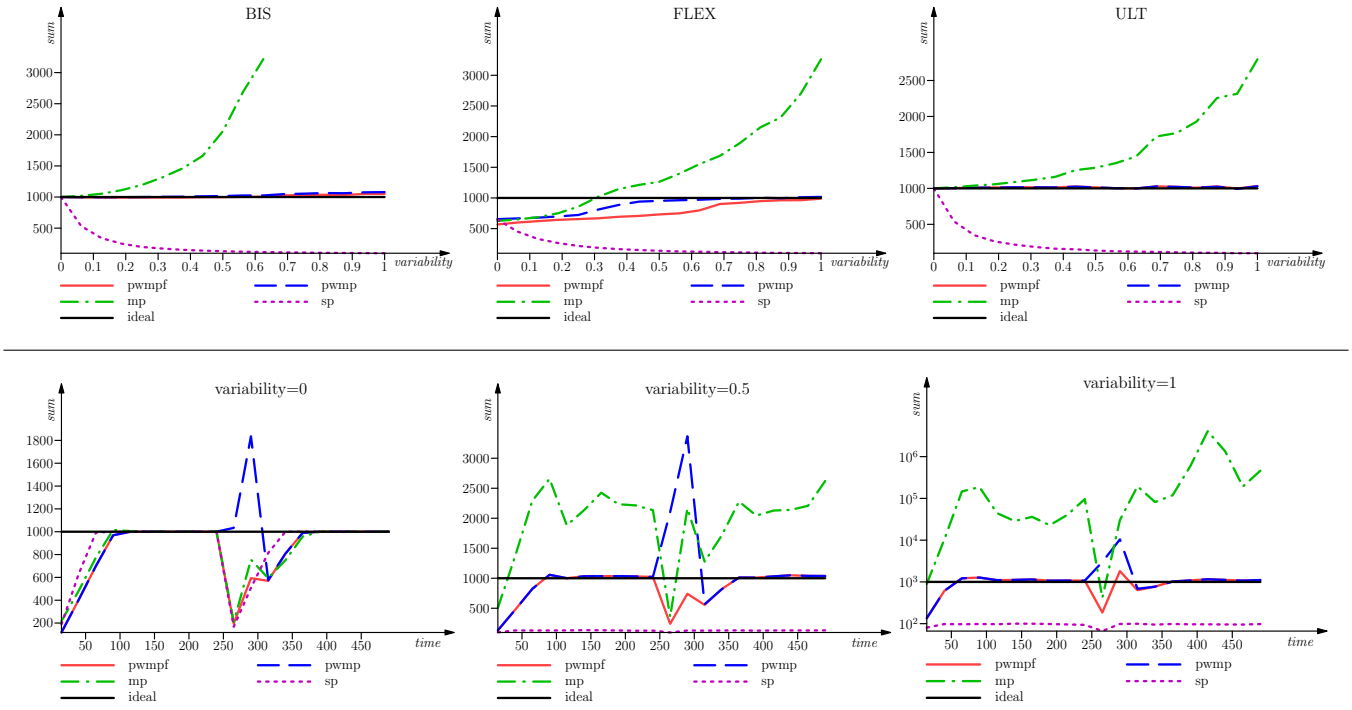

Figure 2: Number of devices measured through different aggregation and distance estimation routines, upon increasing variability with a source change at $T = 250s$. Evolution through time is presented for variabilities 0, 0.5, 1 (bottom) and BIS gradient, together with the average count in the whole time window $0 - 500s$ for continuously increasing variabilities (top) and different gradient routines.

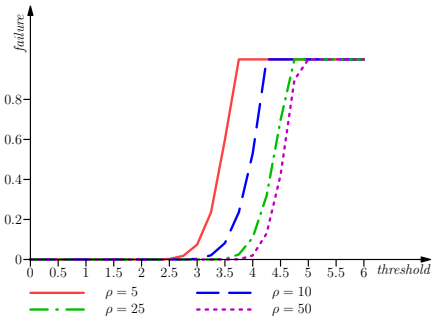

Figure 3: Probability of transmission failure over 1000 hops in idempotent $C_{\mathbf{pwmp}}$ under increasing thresholds and device density ρ (as number of devices over an R^2 area where R is the communication radius).

the first condition to hold, values need to decrease (in average) as time passes. For the second, values need to increase (in average) as distance from the source increases. We thus chose (positive) v_δ according to the simplest formula satisfying the two conditions:

$$v_\delta = d(\delta) + (500 - t(\delta))v$$

where $d(\delta)$ is the distance from the source as computed by a given distance estimation algorithm, $t(\delta)$ is the time elapsed from the start of the simulation and $v = 1m/s$ is an arbitrarily chosen conversion constant. The first term of the formula addresses the impact of distant values, while the second term ensures that obsolete values are similarly significant (with the additive constant 500 so that $v_\delta > 0$ for plotting convenience). Note that the proposed formula for v_δ is archetypal of (more involvedly defined) similar problems that may naturally occur in practical applications: for example, v_δ could track the progress of shipments moving at speed v from warehouses (at distances $d(\delta)$) towards the source, which all started at $t = 0$ so that the maximum v_δ represents the farthest away item.

We run 20 instances of each scenario and averaged the results, which had relative standard error below 5% for all variabilities and FLEX or BIS distance estimations, below 12% for ULT distance estimation. Figure 4 summarises the evaluation results.

In these simulations, the BIS distance estimation algorithm is consistently better regardless of the collection strategy used, followed by FLEX and ULT: FLEX fails to produce consistent results for the v_δ values after the source switch, while ULT does not allow the collection algorithms to stabilise. As before, single-path collection systematically underestimates the ideal value for non-zero degrees of variability, while multi-path collection overestimates the value by not being able to noticeably react to input changes even in fully-static environments (see Section 2.2).

On the other hand, C_{pwmp} collection always significantly outperforms the other strategies. The addition of flow control has a negligible impact, and thus is not shown in the graphs. The best overall results are again obtained by the C_{pwmp} collection with

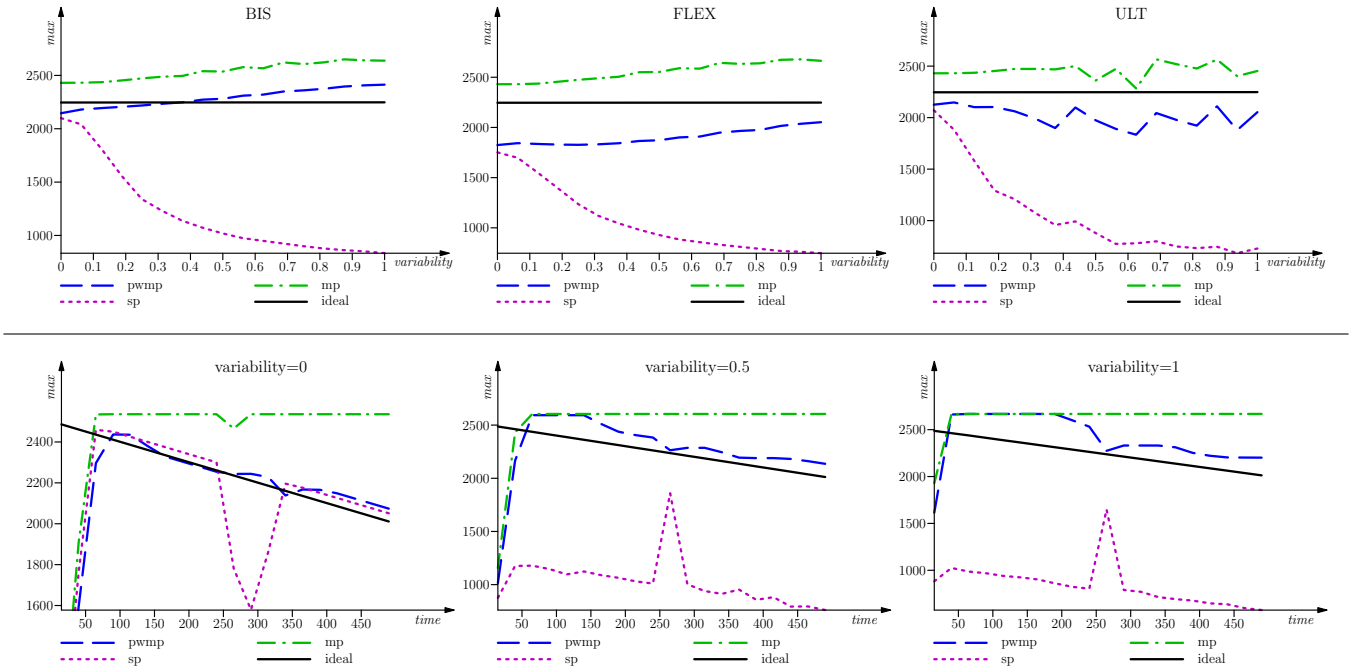

Figure 4: Progress tracking through different aggregation and distance estimation routines, upon increasing variability with a source change at $T = 250s$. Evolution through time is presented for variabilities 0, 0.5, 1 (bottom) and BIS gradient, together with the average count in the whole time window $0 - 500s$ (top) and different gradient routines.

flow control and BIS distance estimation, which has a relative error below 10% even for the highest values of variability.

4.3 Case Study: Crowd Safety Service

Finally, we tested the performance of the different collection algorithms in a realistic case study of crowd safety services on real GPS traces. In fact, high concentrations of people in constrained areas have the simultaneous effects of *(i)* overwhelming traditional wireless infrastructures, and *(ii)* creating zones of dangerous overcrowding, where even small incidents can induce stampedes, injuries or deaths [2]. As the number of IoT-like connected devices increases, however, distributed crowd monitoring can be more effective even in absence of centrally deployed infrastructures, thus allowing for effective prevention of accidents and stampedes.

We built a crowd safety service upon previously developed ones [9], basing on simple conservative estimates of dangerous crowding via level of service (LoS) ratings [17], with LoS D (> 1.08 people$/m^2$) denoting an area at risk of overcrowding (*risk* area) and LoS E (> 2.17 people$/m^2$) in a group of at least 1000 people indicating a potentially dangerous density (*overcrowded* area). We estimated local people density through the formula $\rho = |N| / (p\pi r^2 w)$, where $|N|$ is the count of neighbours within range $r = 30m$, $p = 2\%$ estimates the proportion of people with a device running the app, and $w = 0.25$ estimates the fraction of walkable space in the local urban environment. The *risk* area was computed locally, whereas the *overcrowded* area was determined by averaging local densities over contiguous regions through the collection blocks (since LoS E requires the density to be measured over at least 1000 people).

We considered people who were within $100m$ of an *overcrowded* area in the last $100s$ to be warned of the risk and possibly steered (*warning* area). Warned people were given dispersing directions calculated based on a physical model, associating an equal repulsive force field to each close neighbour. We placed 6000 mobile personal devices with a $100m$ connection range and $20s$ average computation interval (low consumption settings), each following one of the 1500 available[4] GPS traces of people's smartphones for the 2013 Vienna City Marathon, a benchmark scenario considered in [9, 35] among many. We tested scenarios with varying probabilities $P_{\text{follows}} = 0, 0.25$ for an application user to actually follow steering instructions given by the smartphone. We run 20 instances of each scenario and averaged the results, which had a 11% relative standard error between them. Figure 5 summarises the evaluation results.

Scenarios with $P_{\text{follows}} = 0$ were included to compare the performance of different algorithms in measuring overcrowding under identical conditions. In these scenarios, multi-path and C_{pwmp} performed almost identically, showing a somewhat steady course getting slightly worse towards the end. On the other hand, single-path collection was less effective in measuring overcrowded and warning areas, as the averaging routine wasn't able to gather distant values and occasionally underestimated the situation. Thus, the evaluation of single-path on $P_{\text{follows}} = 0.25$ is less reliable as well and cannot be directly compared to that of the other algorithms. In scenarios with $P_{\text{follows}} = 0.25$, C_{pwmp} reduced the overcrowded area slightly

[4]Due to the very low density of available traces, the resulting network is highly disconnected and the advantages of our algorithm are minimised, although still visible.

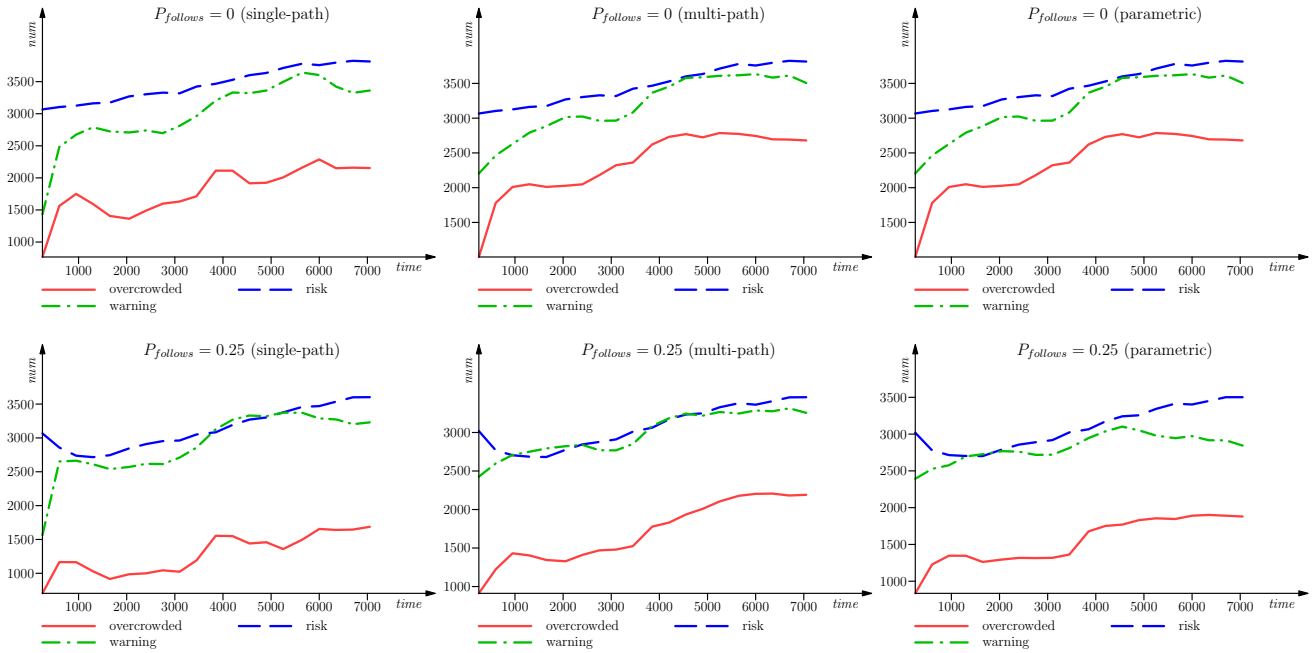

Figure 5: Crowd tracking through different aggregation routines—users' probabilities of following dispersion advices equal to 0 and 0.25. Number of people belonging to the selected LoS areas is shown during an interval of 2 hours (7200s).

more than multi-path, while warning less people towards the end (thus reducing the obtrusiveness of the application).

5 CONCLUSIONS

We introduced *parametric weighted multi-path* collection C_{pwmp}, a new algorithm for summarising distributed data improving over state-of-the-art implementations of the C block. Experimental evaluation in isolation tests shows that C_{pwmp} achieves adequate accuracy even in very high-variability scenarios where other approaches are infeasible, both in arithmetic (device counting) and idempotent (progress tracking) aggregations. Namely, C_{pwmp} paves the way for effective exploitation of fully-distributed collection processes – hence, distributed situation recognition – in mobile multi-agent systems, e.g., when agent communication is carried on by wireless broadcasts.

Evaluation in a realistic case study of crowd safety services on real GPS traces shows a consistent improvement, though also suggests that more significant improvements may be possible in future settings with higher densities of devices. Hence, in future works, ad-hoc adaptations of the algorithm may be developed to improve its performance, in particular by reducing the error peaks that are still present in response to input discontinuities. In these situations, potential-descending data flows may create loops leading to exponential increases in error, so that time-driven strategies for preventing them would be significantly beneficial. Furthermore, the weights given to links may be more accurately tuned through statistical methods, given information on the average movement of devices, improving accuracy also in steady state.

C_{pwmp} is an algorithm that effectively abstracts from the heterogeneous nature of devices, assuming they all have same interaction/computation capabilities. When heterogeneity play a crucial role, one needs to adapt it to the situation at hand. Nodes that need not participate in collection, must simply be excluded (by the underlying platform or at application level) from the "neighbouring" relation upon which the algorithm is devised. Future works will consider that case in which devices have dynamic availability of computational resources, and task allocation has to be considered to select which device has to be used to collaborate on collection, and to which extent.

As the notion of field is shown to be a useful one to model and implement distributed collection processes, a future work will be to explore how it can conversely be used for planning and action [33]. Early ideas are presented in [32] where all components of a MAPE loop (Monitor, Analyse, Plan and Act) are supported by field computations: generally, specific building blocks need to be implemented to streamline smooth development of advanced adaptive behaviour.

Finally, we plan to integrate collection, and other key self-organisation building blocks [31], as services available to intelligent agent platforms like, e.g., Jason [12], also to investigate theory and practice of distributed notions of knowledge, plans and goals.

ACKNOWLEDGEMENTS

We thank the anonymous AAMAS reviewers for the helpful feedback received.

REFERENCES

[1] Marco Aldinucci, Stefano Bagnasco, Stefano Lusso, Paolo Pasteris, Sara Vallero, and Sergio Rabellino. 2016. The Open Computing Cluster for Advanced data Manipulation (OCCAM). In *The 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP)*. San Francisco, USA.

[2] Bernhard Anzengruber, Danilo Pianini, Jussi Nieminen, and Alois Ferscha. 2013. Predicting Social Density in Mass Events to Prevent Crowd Disasters. In *Social Informatics 5th International Conference (SocInfo 2013), Proceedings*. 206–215.

[3] Giorgio Audrito and Sergio Bergamini. 2017. Resilient Blocks for Summarising Distributed Data. In *Proceedings First Workshop on Architectures, Languages and Paradigms for IoT (ALP4IoT)*. 23–26.

[4] Giorgio Audrito, Roberto Casadei, Ferruccio Damiani, and Mirko Viroli. 2017. Compositional Blocks for Optimal Self-Healing Gradients. In *Self-Adaptive and Self-Organizing Systems (SASO), 2017*. IEEE, 91–100.

[5] Giorgio Audrito, Ferruccio Damiani, and Mirko Viroli. 2017. Optimally-Self-Healing Distributed Gradient Structures Through Bounded Information Speed. In *COORDINATION 2017 (LNCS)*, Vol. 10319. Springer, 59–77. Best paper at Coordination 2017.

[6] Jacob Beal. 2009. Flexible Self-healing Gradients. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*. ACM, 1197–1201.

[7] Jacob Beal, Stefan Dulman, Kyle Usbeck, Mirko Viroli, and Nikolaus Correll. 2013. Organizing the Aggregate: Languages for Spatial Computing. In *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*, Marjan Mernik (Ed.). IGI Global, Chapter 16, 436–501.

[8] Jacob Beal, Olivier Michel, and Ulrik Pagh Schultz. 2011. Spatial Computing: Distributed Systems That Take Advantage of Our Geometric World. *TAAS* 6, 2 (2011), 11:1–11:3. https://doi.org/10.1145/1968513.1968514

[9] Jacob Beal, Danilo Pianini, and Mirko Viroli. 2015. Aggregate Programming for the Internet of Things. *IEEE Computer* 48, 9 (2015), 22–30.

[10] Jacob Beal, Kyle Usbeck, Joseph Loyall, Mason Rowe, and James Metzler. 2018. Adaptive Opportunistic Airborne Sensor Sharing. *ACM Trans. Auton. Adapt. Syst.* 13, 1, Article 6 (April 2018), 29 pages.

[11] Nicola Bicocchi, Marco Mamei, and Franco Zambonelli. 2012. Self-organizing virtual macro sensors. *TAAS* 7, 1 (2012), 2:1–2:28.

[12] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldrige. 2007. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons.

[13] R. Claes and T. Holvoet. 2014. Traffic Coordination Using Aggregation-Based Traffic Predictions. *IEEE Intelligent Systems* 29, 4 (July 2014), 96–100. https://doi.org/10.1109/MIS.2014.73

[14] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. 2005. Context is Key. *Commun. ACM* 48, 3 (March 2005), 49–53. https://doi.org/10.1145/1047671.1047703

[15] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[16] Jose Luis Fernandez-Marquez, Giovanna Di Marzo Serugendo, Sara Montagna, Mirko Viroli, and Josep Lluís Arcos. 2013. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing* 12, 1 (2013), 43–67.

[17] John J Fruin. 1971. *Pedestrian planning and design*. Technical Report. Metropolitan Assoc. of Urban Designers and Environmental Planners.

[18] Jomi Fred Hübner, Olivier Boissier, Rosine Kitio, and Alessandro Ricci. 2010. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems* 20, 3 (2010), 369–400. https://doi.org/10.1007/s10458-009-9084-y

[19] Qingzhi Liu, Andrei Pruteanu, and Stefan Dulman. 2013. Gradient-Based Distance Estimation for Spatial Computers. *Comput. J.* 56, 12 (2013), 1469–1499.

[20] Marco Mamei, Franco Zambonelli, and Letizia Leonardi. 2002. Co-Fields: Towards a Unifying Approach to the Engineering of Swarm Intelligent Systems. In *Engineering Societies in the Agents World III, Third International Workshop, ESAW 2002, Madrid, Spain, September 16-17, 2002, Revised Papers (Lecture Notes in Computer Science)*, Paolo Petta, Robert Tolksdorf, and Franco Zambonelli (Eds.), Vol. 2577. Springer, 68–81. https://doi.org/10.1007/3-540-39173-8_6

[21] Marco Mamei, Franco Zambonelli, and Letizia Leonardi. 2004. Co-Fields: A Physically Inspired Approach to Motion Coordination. *IEEE Pervasive Computing* 3, 2 (2004), 52–61. https://doi.org/10.1109/MPRV.2004.1316820

[22] Hassnaa Moustafa and Yan Zhang. 2009. *Vehicular Networks: Techniques, Standards, and Applications* (1st ed.). Auerbach Publications, Boston, MA, USA.

[23] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. 2008. Synopsis diffusion for robust aggregation in sensor networks. *TOSN* 4, 2 (2008), 7:1–7:40.

[24] Danilo Pianini, Sara Montagna, and Mirko Viroli. 2013. Chemical-oriented simulation of computational systems with ALCHEMIST. *J. Simulation* 7, 3 (2013), 202–215.

[25] Danilo Pianini, Mirko Viroli, and Jacob Beal. 2015. Protelis: Practical Aggregate Programming. In *ACM SAC 2015*. 1846–1853.

[26] Raffaele Quitadamo and Franco Zambonelli. 2008. Autonomic communication services: a new challenge for software agents. *Autonomous Agents and Multi-Agent Systems* 17, 3 (2008), 457–475.

[27] Alessandro Ricci, Michele Piunti, and Mirko Viroli. 2011. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23, 2 (2011), 158–192. https://doi.org/10.1007/s10458-010-9140-7

[28] Giovanna Di Marzo Serugendo, Marie Pierre Gleizes, and Anthony Karageorgos. 2005. Self-organization in multi-agent systems. *Knowledge Eng. Review* 20, 2 (2005), 165–189. https://doi.org/10.1017/S0269888905000494

[29] Ajay K Talele, Suraj G Patil, and Nilkanth B Chopade. 2015. A survey on data routing and aggregation techniques for wireless sensor networks. In *Pervasive Computing (ICPC), 2015 International Conference on*. IEEE, 1–5.

[30] László Z. Varga. 2018. Two Prediction Methods for Intention-Aware Online Routing Games. In *Multi-Agent Systems and Agreement Technologies*, Francesco Belardinelli and Estefanía Argente (Eds.). Springer International Publishing, Cham, 431–445.

[31] Mirko Viroli, Giorgio Audrito, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. 2018. Engineering Resilient Collective Adaptive Systems by Self-Stabilisation. *ACM Trans. Model. Comput. Simul.* 28, 2, Article 16 (March 2018), 28 pages.

[32] Mirko Viroli, Antonio Bucchiarone, Danilo Pianini, and Jacob Beal. 2016. Combining Self-Organisation and Autonomic Computing in CASs with Aggregate-MAPE. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), Augsburg, Germany, September 12-16, 2016*, Sameh Elnikety, Peter R. Lewis, and Christian Müller-Schloer (Eds.). IEEE, 186–191. https://doi.org/10.1109/FAS-W.2016.49

[33] Mirko Viroli, Danilo Pianini, Alessandro Ricci, and Angelo Croatti. 2017. Aggregate plans for multiagent systems. *International Journal of Agent-Oriented Software Engineering* 4, 5 (2017), 336–365.

[34] D. Ye, M. Zhang, and A. V. Vasilakos. 2017. A Survey of Self-Organization Mechanisms in Multiagent Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47, 3 (March 2017), 441–461. https://doi.org/10.1109/TSMC.2015.2504350

[35] Franco Zambonelli, Andrea Omicini, Bernhard Anzengruber, Gabriella Castelli, Francesco L. De Angelis, Giovanna Di Marzo Serugendo, Simon A. Dobson, Jose Luis Fernandez-Marquez, Alois Ferscha, Marco Mamei, Stefano Mariani, Ambra Molesini, Sara Montagna, Jussi Nieminen, Danilo Pianini, Matteo Risoldi, Alberto Rosi, Graeme Stevenson, Mirko Viroli, and Juan Ye. 2015. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing* 17 (2015), 236–252.

[36] Ying Zhang, Xuemin Lin, Yidong Yuan, Masaru Kitsuregawa, Xiaofang Zhou, and Jeffrey Xu Yu. 2010. Duplicate-Insensitive Order Statistics Computation over Data Streams. *IEEE Trans. Knowl. Data Eng.* 22, 4 (2010), 493–507.