Engineering Swarm Behaviours through (Hybrid) Aggregate Computing

Gianluca Aguzzi gianluca.aguzzi@unibo.it

Alma Mater Studiorum – Università di Bologna

08/04/2024

Outline

1. Swarm Behaviors – Overview
2. Aggregate Computing Toolchain
   - ScaFi
   - MacroSwarm
3. Hybrid Aggregate Computing
   - Application-level
   - Runtime-level

Swarm Behaviors

Definition

A *swarm behavior* a collective behavior that emerges from the interaction of a group of agents.

Taxonomy

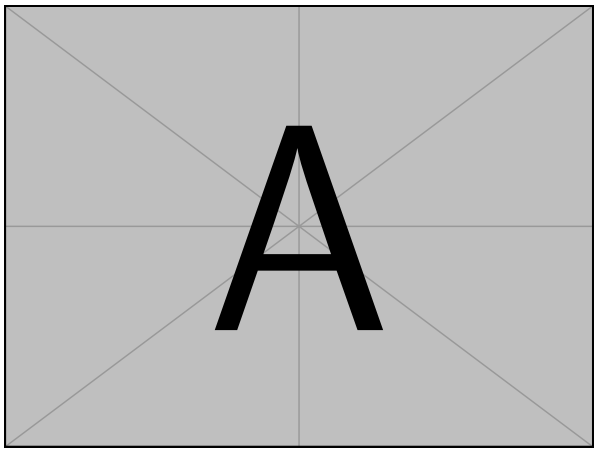
From a swarm robotics perspective, swarm behaviors can be classified into three main categories [1]:

- *Spatial-organization behaviors*: the agents organize themselves in a specific spatial configuration
- *Navigation behaviors*: the agents collectively navigate in the environment following a specific strategy
- *Collective Decision-Making*: the agents collectively make decisions based on the information they have

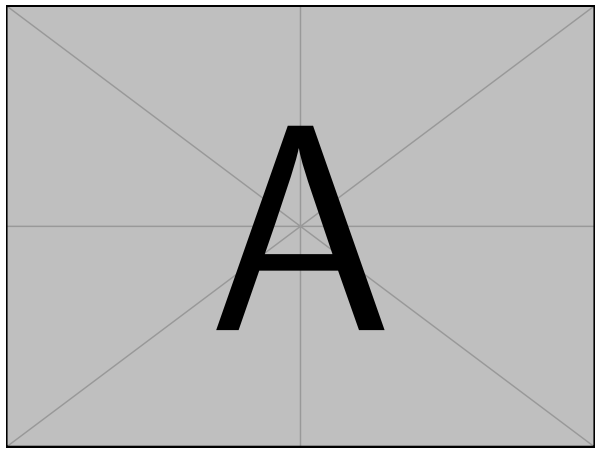
---

[1] M. Brambilla, E. Ferrante, M. Birattari, *et al.*, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, 2013. doi: 10.1007/S11721-012-0075-2. [Online]. Available: https://doi.org/10.1007/s11721-012-0075-2

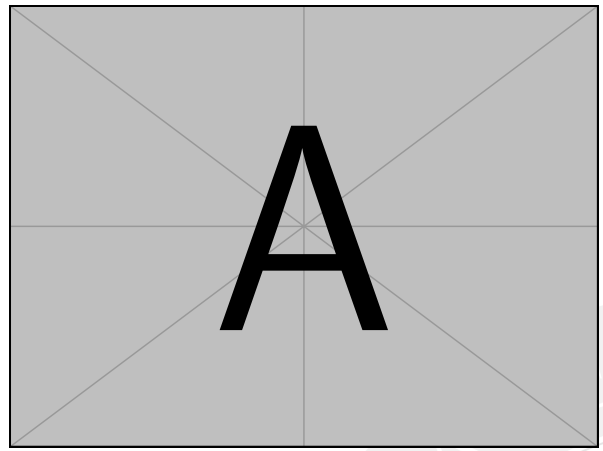
Spatial-organization behaviors



Aggregation

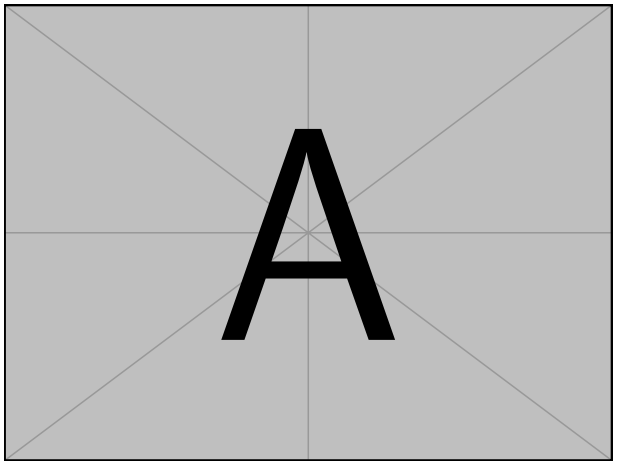


Dispersion

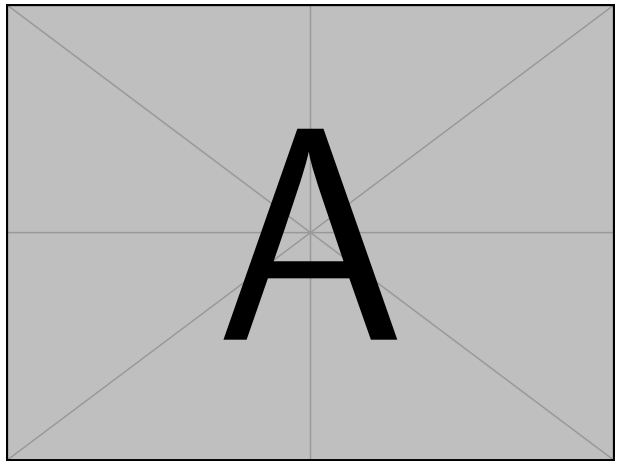


Flocking

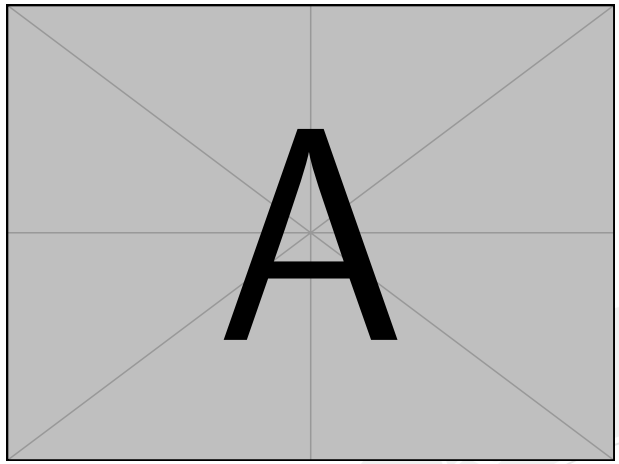
Navigation behaviors



Exploration

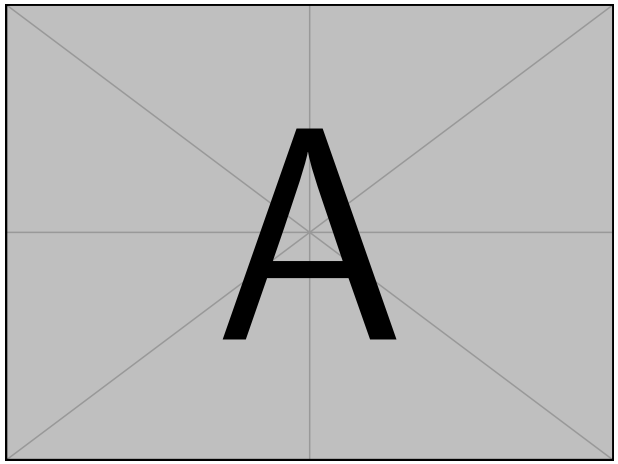


Collective Motion

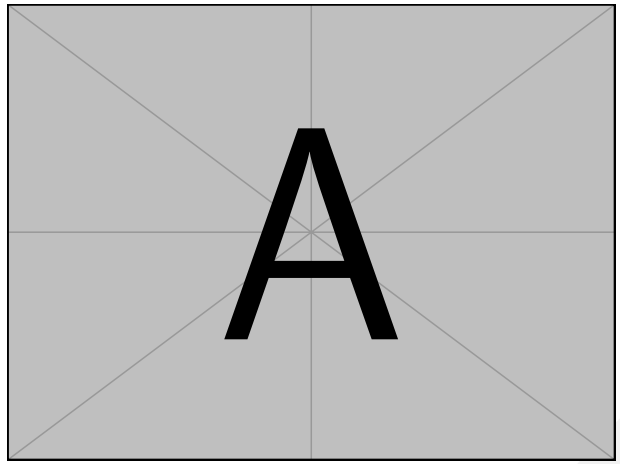


Group Transportation

Collective Decision-Making



Exploration



Collective Motion

Engineering Swarm Behaviors I

Definition

Engineering swarm behaviors is the process of designing, implementing, and deploying algorithms that enable a group of agents to exhibit a specific collective behavior.

Challenges

- *Scalability*: the collective behavior should scale with the number of agents
- *Robustness*: even in case of failures, the collective behavior should be maintained
- *Flexibility*: the collective behavior should be easily adaptable to different scenarios
- *Efficiency*: the collective behavior should be efficient in terms of energy and time

Engineering Swarm Behaviors II

Properties

- *Emergence*: the collective behavior emerges from the interaction of the agents
- *Decentralization*: the collective behavior is achieved without a central controller
- *Self-organization*: the collective behavior is achieved through local interactions
- *Homogeneity*: the agents are homogeneous and have the same capabilities
- *Limited Communication*: the agents have limited communication capabilities

State-of-the-art

- *manual design*: the collective behavior is manually designed through
  - bio-inspired algorithms, macroprogramming languages ➔ aggregate programming
- *automatic design*: the collective behavior is automatically designed through machine learning techniques
  - reinforcement learning, evolutionary algorithms, multi-agent reinforcement learning
- ***hybrid design***: a combination of manual and automatic design

Aggregate Programming – What we want

Goal

- a **minimal core** to cover the main functionality of field calculus
- a **set of libraries** that cover the main building blocks
- an **high-level API** for domain-specific applications

Challenges

- Support several kinds of devices
  - *constrained, commodity, consumer, server*
- Support several infrastructures
  - *peer-to-peer, server-based, cloud-edge*
- Expressive and simple enough to reduce the cognitive load

Aggregate programming: current implementations I

A large number of incarnations and implementations have been developed over the years, with different *trade-offs* and *goals*.

External DSL

- **Proto** [2]: first incarnation of FC
  - 👍 *good performance* ➔ *low-level*
  - 👎 *hard to use* ➔ *hard to maintain*
- **Protelis** [3]: an external DSL based on Xtext
  - 👍 *good ergonomics* ➔ *high-level*
  - 👍 *extensive libraries* ➔ *easy to use*
  - 👎 *hard to maintain*
  - 👎 *JVM-based* ➔ *hard to port to constrained devices*

Aggregate programming: current implementations II

Internal DSL

- **FCPP** [4]: a performance-oriented internal DSL in C++
  - 👍 *good performance* ➔ *low-level*
  - 👎 *hard to use* ➔ *bad ergonomics*
- **ScaFi** [5]: a high-level internal DSL in Scala (current reference implementation and focus of this talk)
  - 👍 *good ergonomics* ➔ *high-level*
  - 👍 *extensive libraries* ➔ *easy to use*
  - 👍 *Cross-platform*
  - 👎 Scala Native ➔ *hard to port to constrained devices*

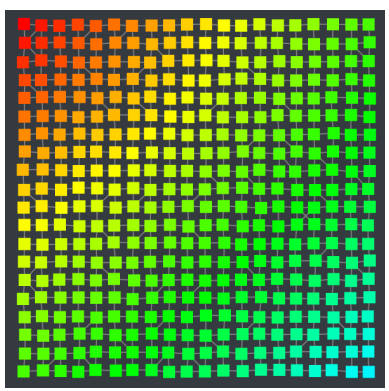
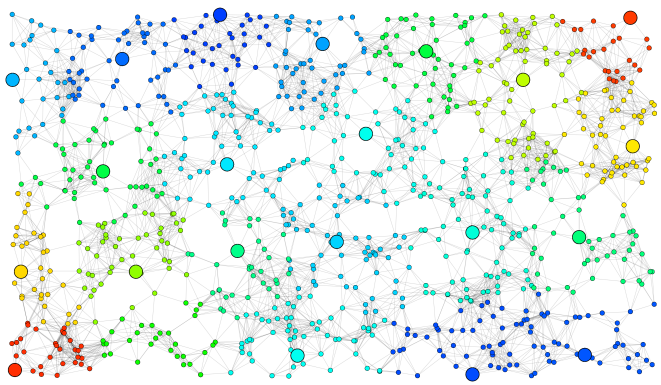
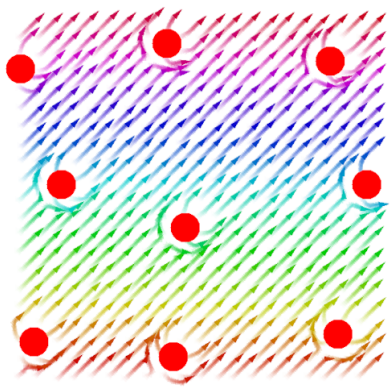
Experimental

- **Collektive**[a]: a high-level DSL in Kotlin leveraging the compiler plugin and the Kotlin Multiplatform
- **RuFi**[b]: a minimal core in Rust (WIP)

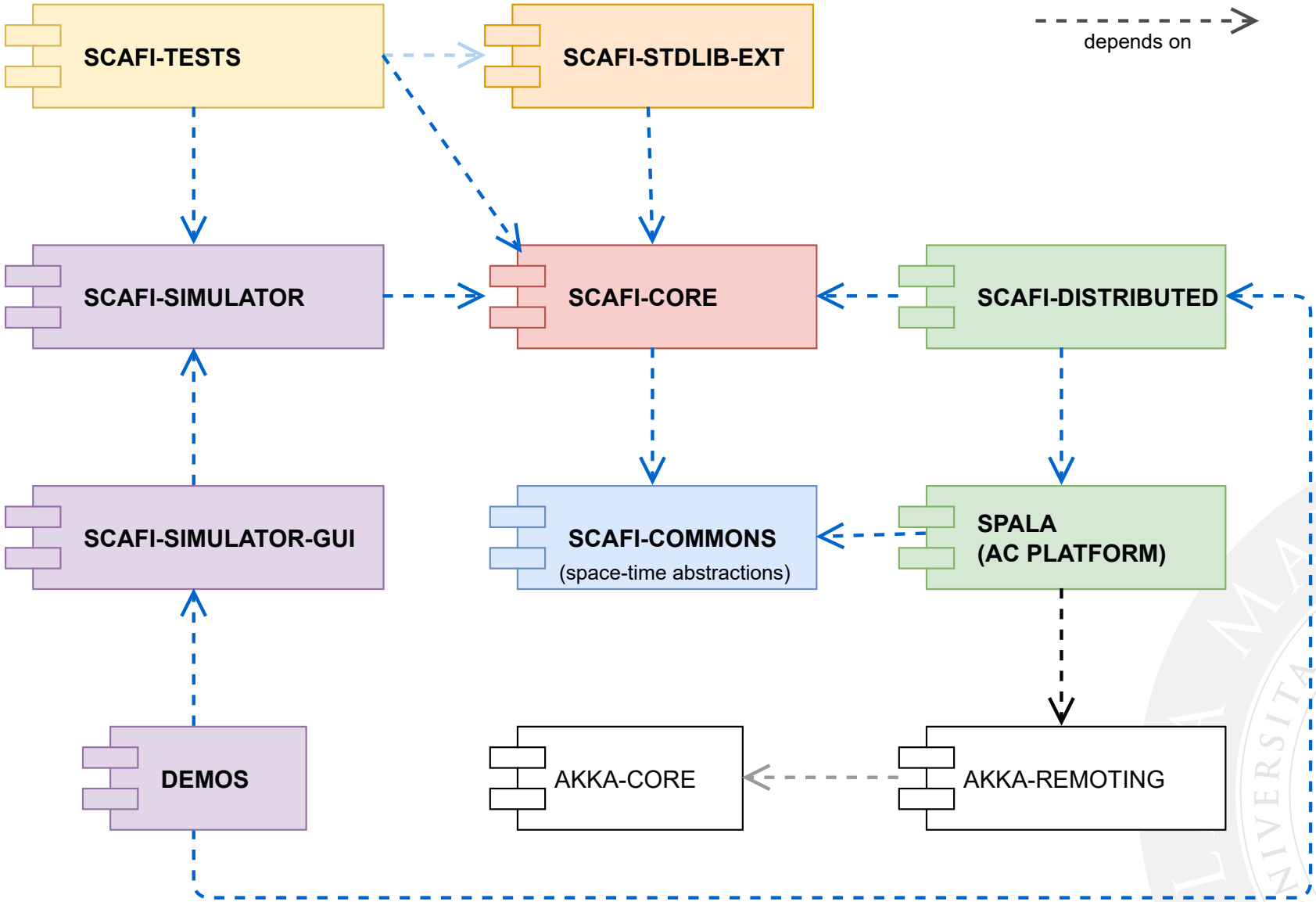
[a] https://github.com/Collektive/collektive
[b] https://github.com/lm98/rufi

ScaFi (Scala Fields)

A Scala toolkit providing an *internal domain-specific* language, *libraries*, a *simulation* environment, and *runtime* support for **practical** aggregate computing systems development

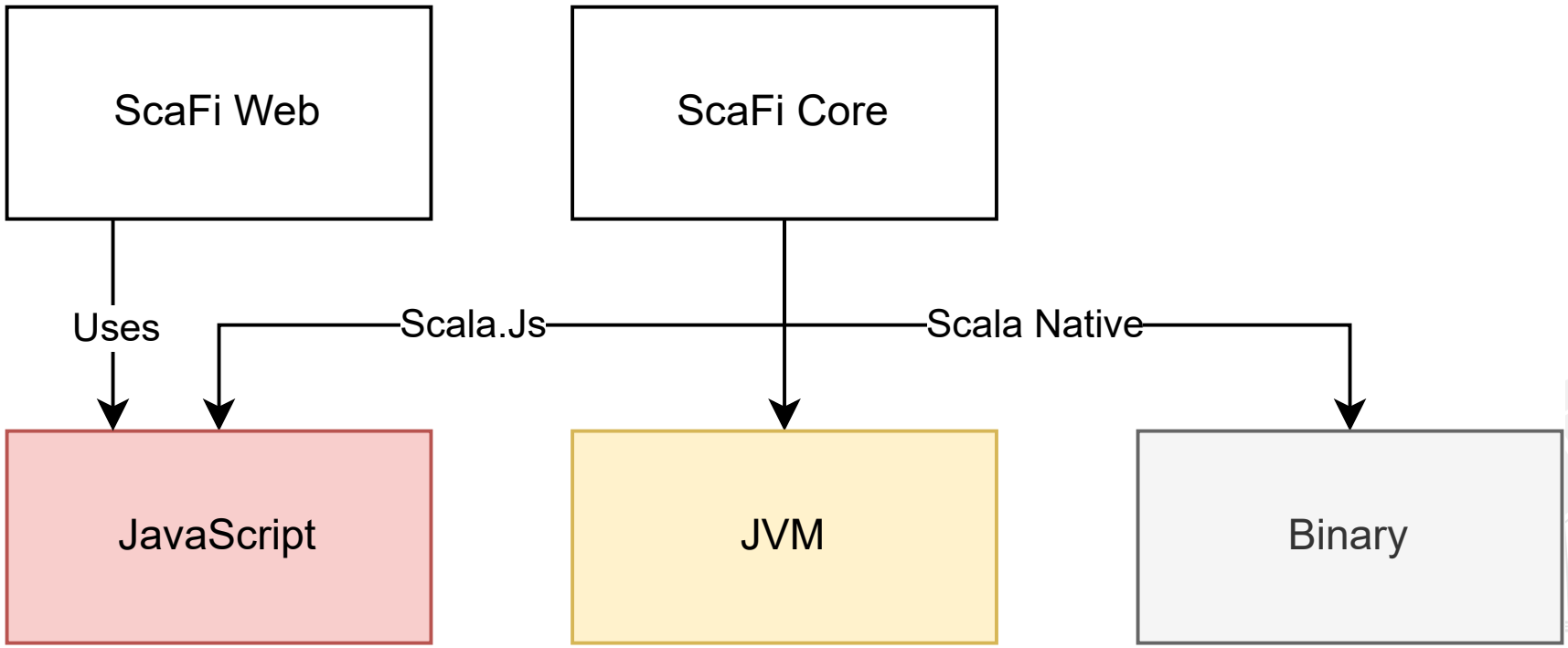
ScaFi: Organization



ScaFi: syntax as a core language/API

```
trait Constructs {
        // the unique identifier of the local device
        def mid(): ID

        // applies fun to the previous result, or to init at the first call
        def rep[A](init: A)(fun: (A) => A): A

        // evaluation of expr at the currently-considered neighbor
        def nbr[A](expr: => A): A

        // accumulates available evaluations of expr, with acc/init monoid
        def foldhood[A](init: => A)(acc: (A,A)=>A)(expr: => A): A

        // splits computation: th where cond is true, el everywhere/time else
        def branch[A](cond: => Boolean)(th: => A)(el: => A): A

        // perception of local sensor
        def sense[A](name: CNAME): A

        // perception of neighbourhood sensor
        def nbrvar[A](name: CNAME): A
        ...
}
```

ScaFi: Cross-platform

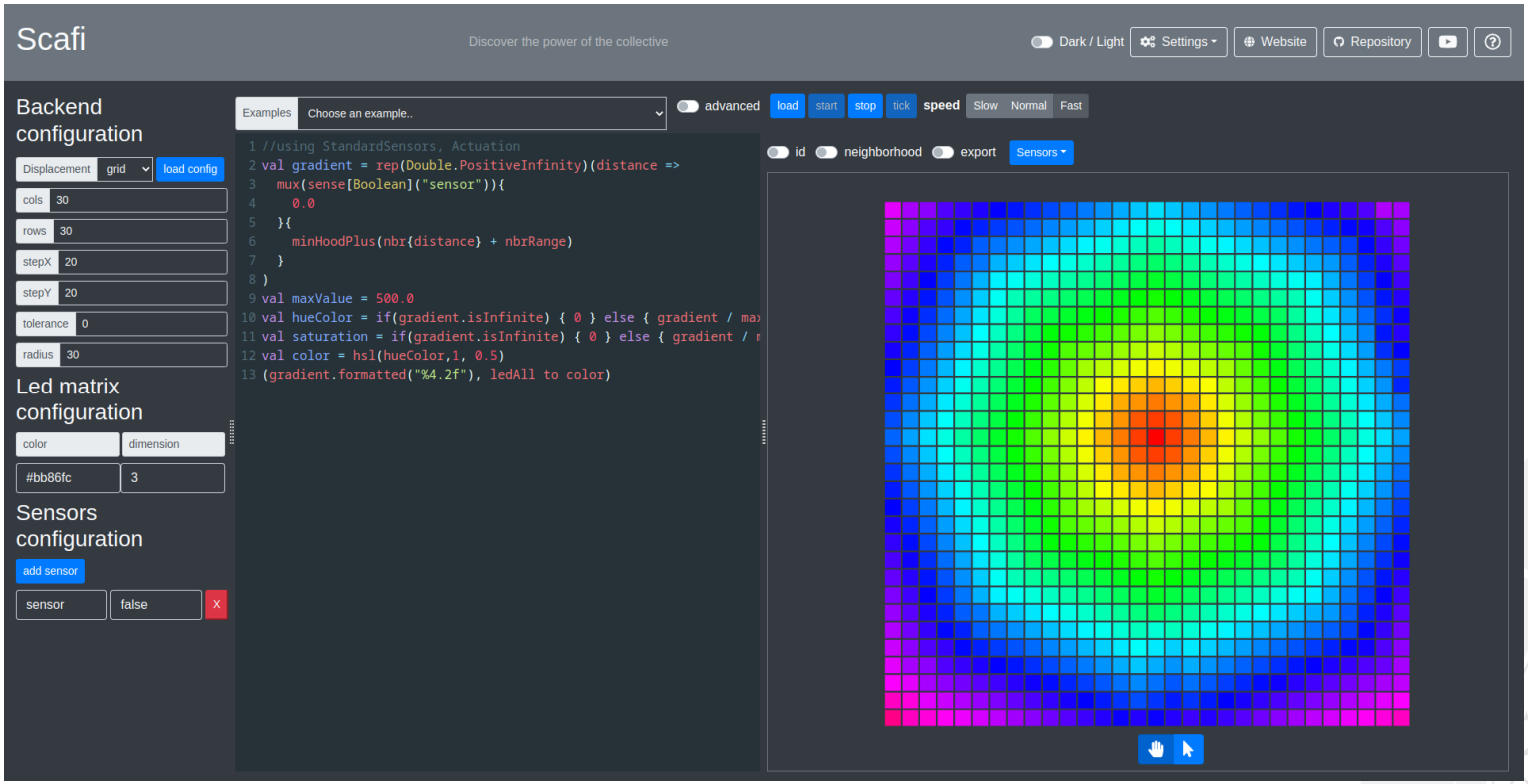


ScaFi: setup an aggregate application

```
package experiments
// STEP 1 Choose an incarnation (simulated or real)
import it.unibo.scafi.incarnations.BasicSimulationIncarnation._

// STEP 2 Define the aggregate program including the right libraries
class MyProgram extends AggregateProgram with Libs {
        // STEP 2.1 Define main logic of the program
        override def main(): Any = rep(0)(_ + 1)
}

// STEP 3 Platform/Node setup (both in simulation/real)

// STEP 3.1 in case of ScaFi simulation:
object SimulationRunner extends Launcher {
  Settings.Sim_ProgramClass = "experiments.MyAggregateProgram"
  Settings.ShowConfigPanel = true
  launch()
}
```

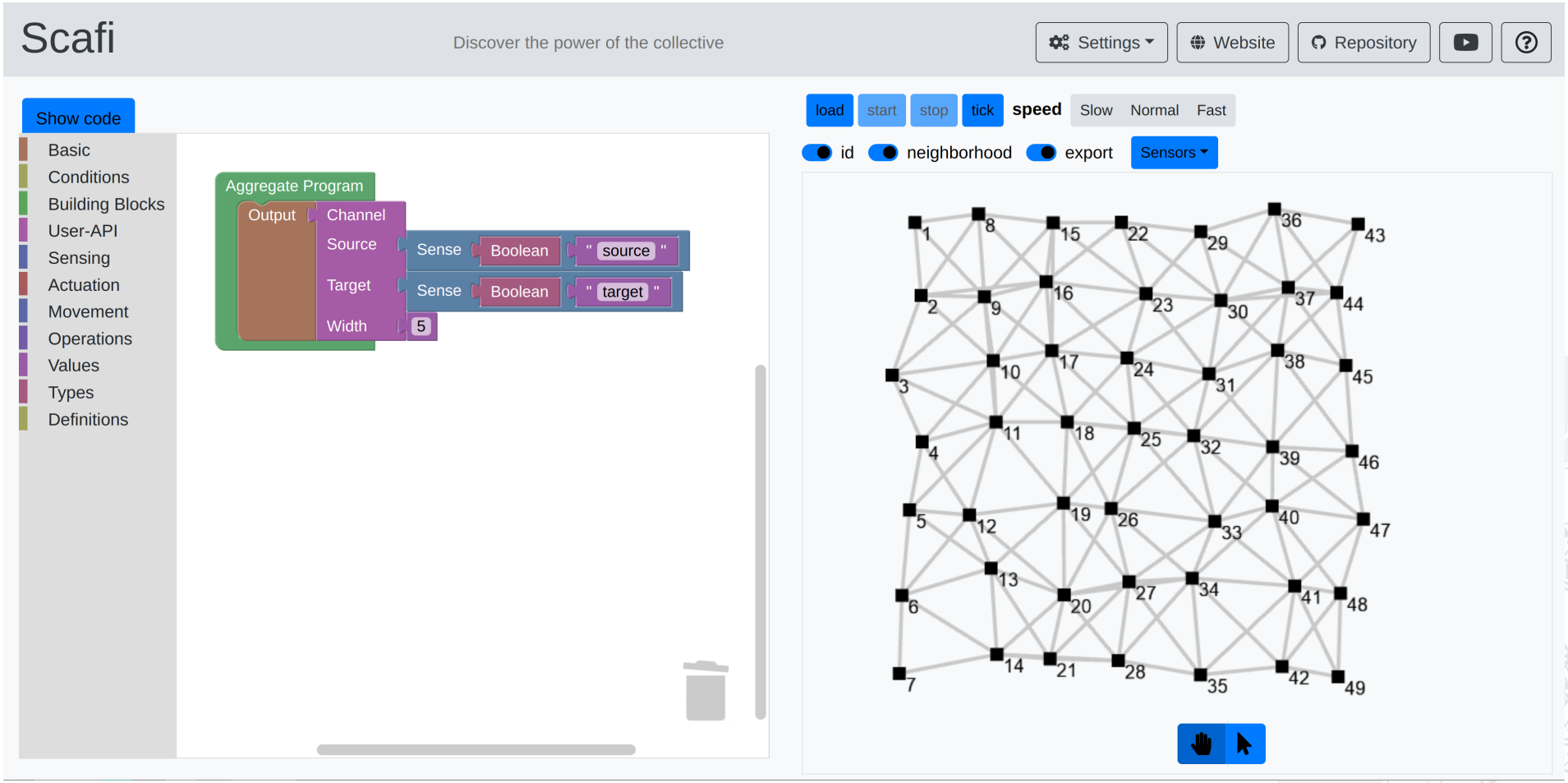
An aggregate computing playground – ScaFi Web!

https://scafi.github.io/web/



ScaFi Blocks

ScaFi Blocks is a visual programming environment for ScaFi.



MacroSwarm

What

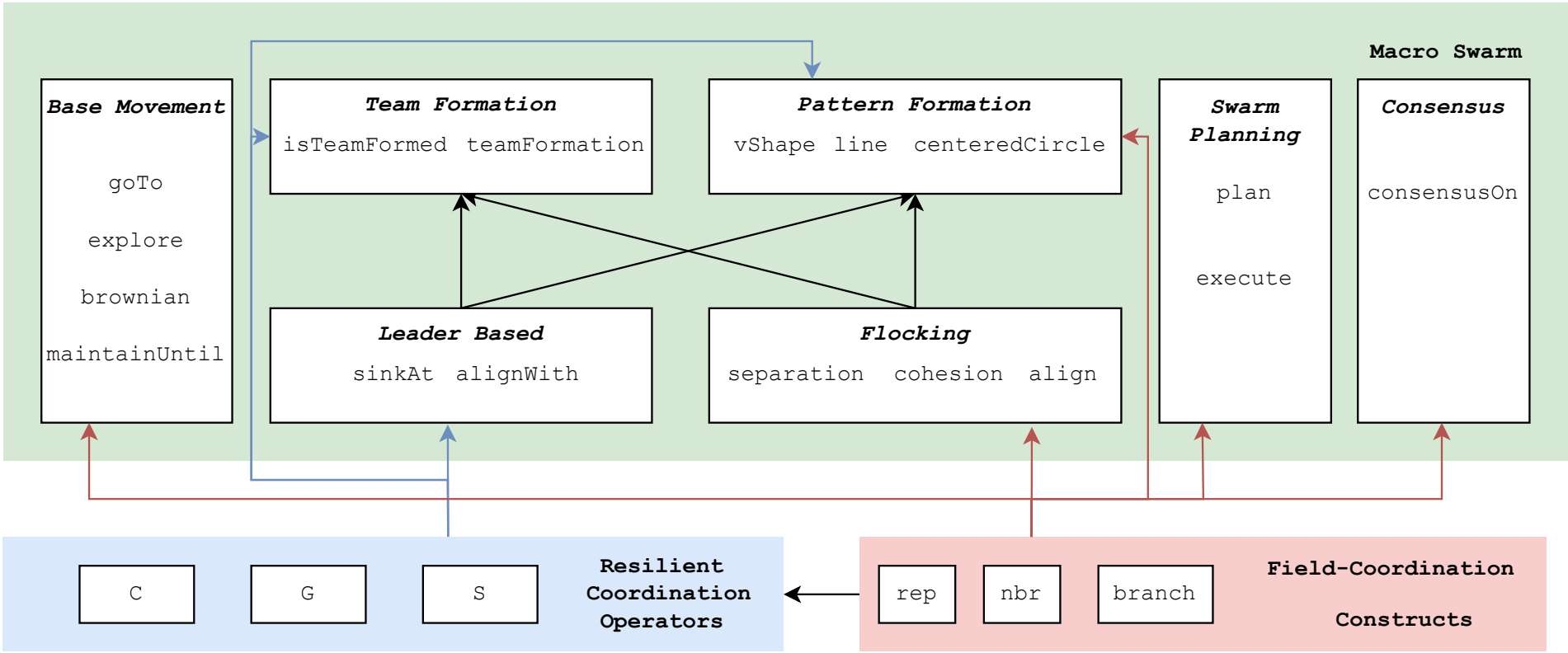
MacroSwarm[a] [6] is a ScaFi library for programming swarm robotics applications.

[a] https://scafi.github.io/macro-swarm/

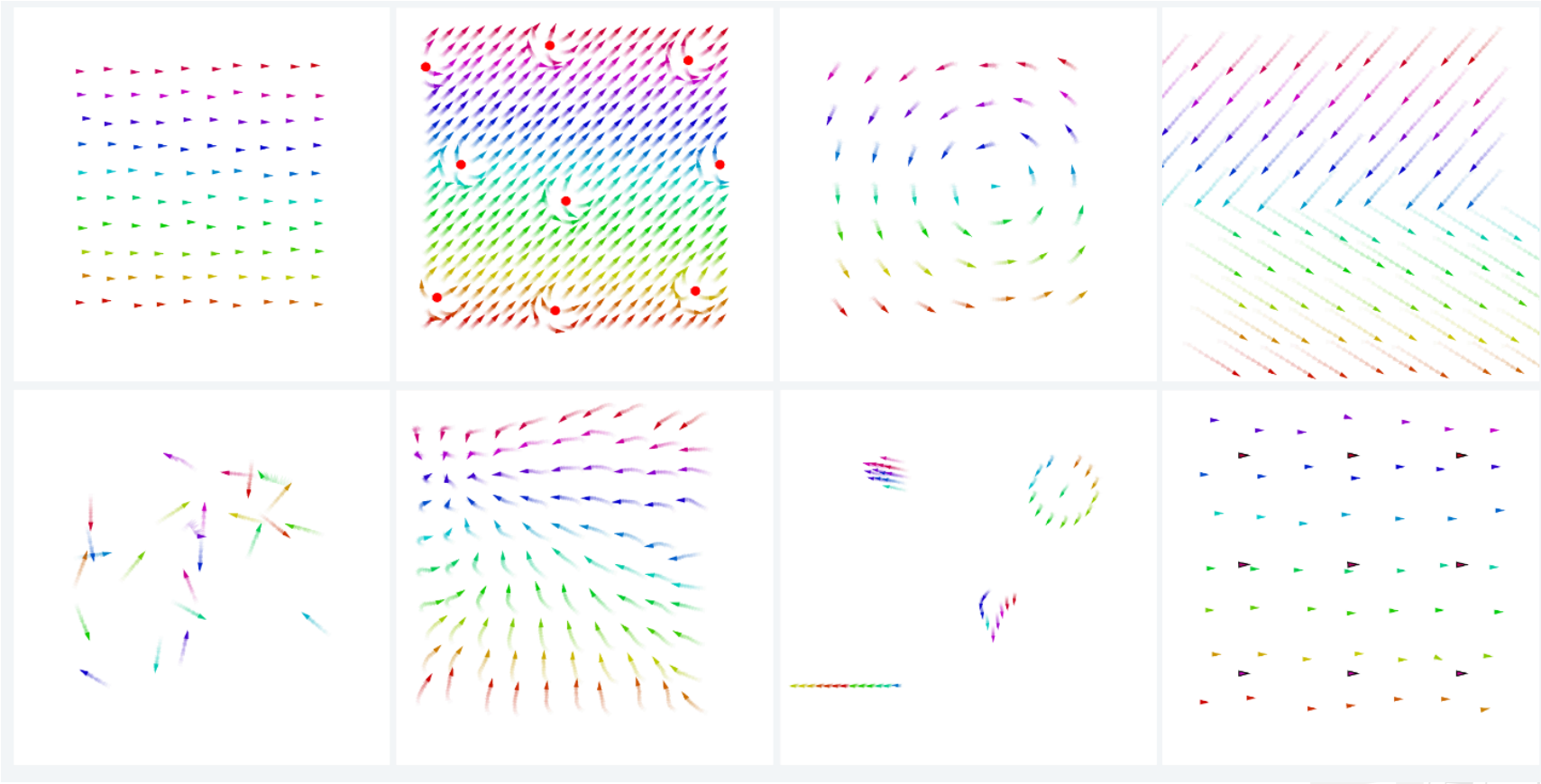
How

- A MacroSwarm program is a function from *sensing* field to *action* field
  - *sensing* field: the information perceived by the agents (e.g., the distance from a source)
  - *action* field: the action to be performed by the agents (e.g., a velocity vector)
- The action field then should be translated into the actual action of the agents based on the specific robot capabilities
  - e.g., a velocity vector can be translated into motor commands

MacroSwarm: API structure



MacroSwarm: Example



Towards Hybrid Aggregate Computing

What

Hybrid Aggregate Computing is the process of combining *manual* (aggregate programming) and *automatic* design techniques to engineer swarm behaviors.

Why

- *Manual design* is good for designing *simple* and *well-understood* behaviors
- *Automatic design* is good for designing *complex* and *unknown* behaviors
- *Hybrid design* is good for combining the best of both worlds

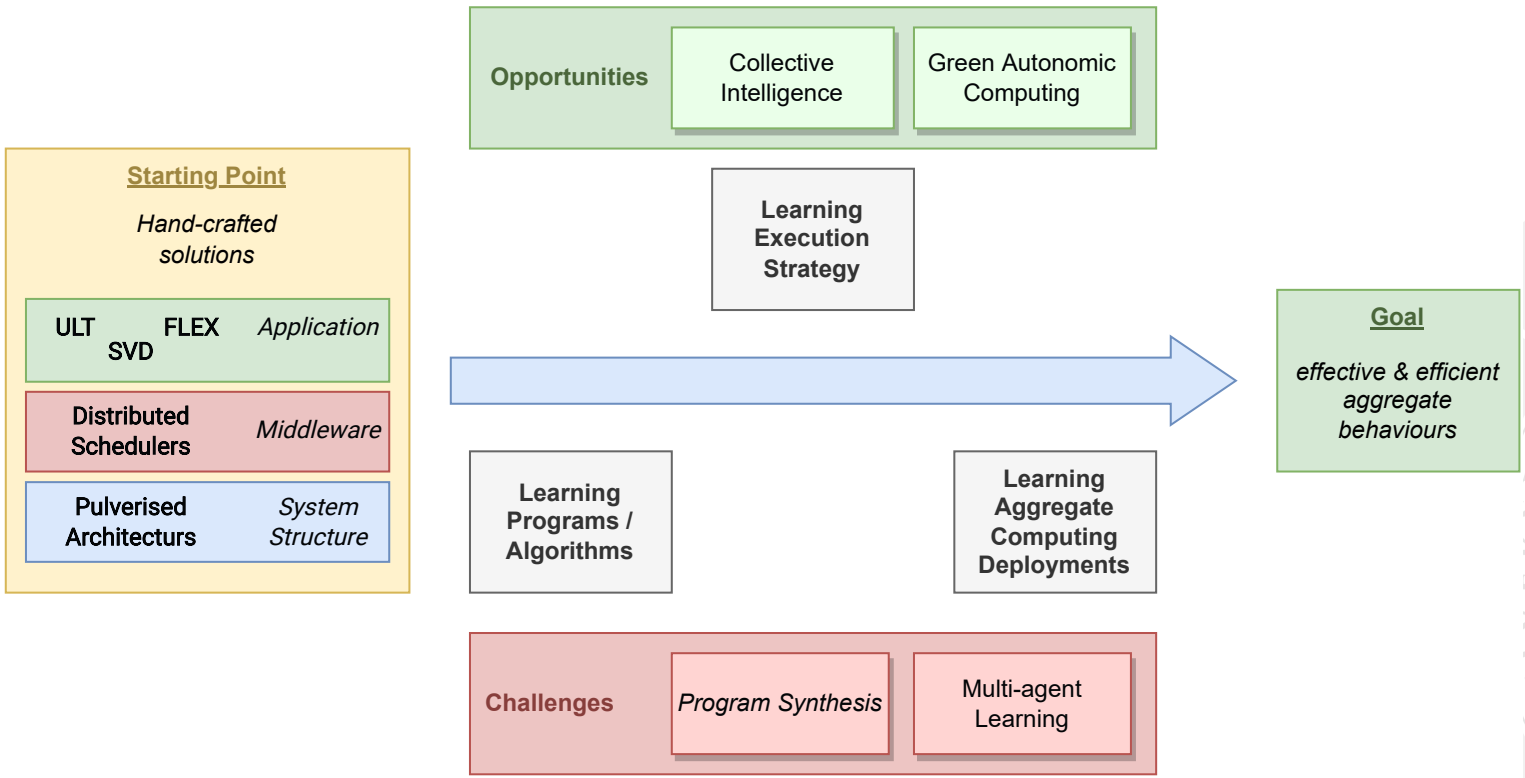
Research Agenda

Goals:

- functionality (e.g., a certain collective behaviour)
- non-functionality (e.g., energy consumption)

Means:

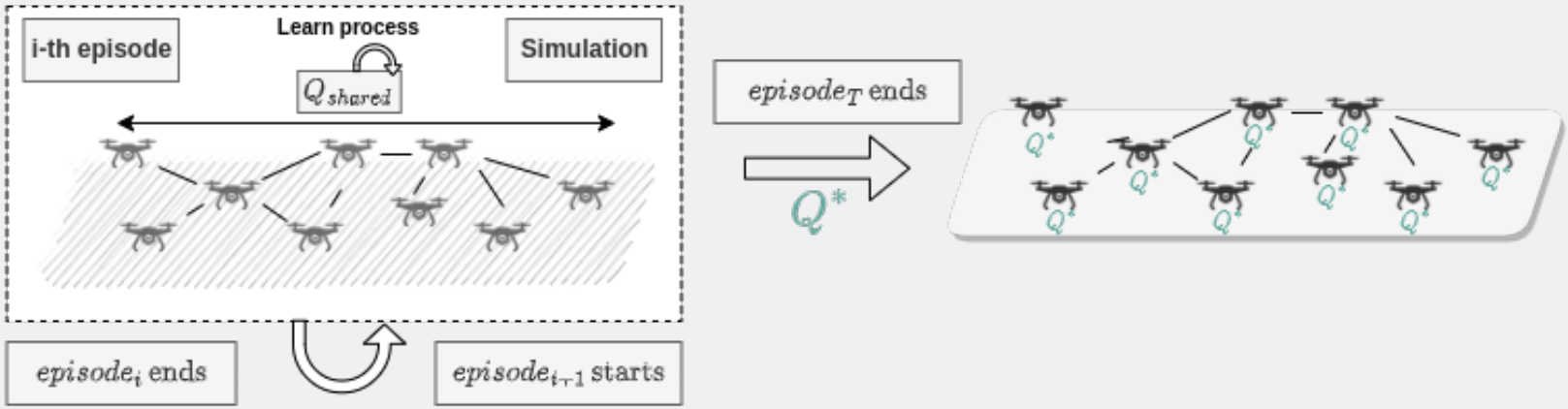
- system structures
- *execution strategy*
- *algorithms*



Hybrid Aggregate Computing – Execution Strategy

Distributed schedulers

- The aggregete computing model does not enforce a global-synchronization
- *Distributed* schedulers [7]: *local* decisions on the *rounds* of the aggregate program
- Idea: *learning* the *best* scheduling strategy directly by-doing

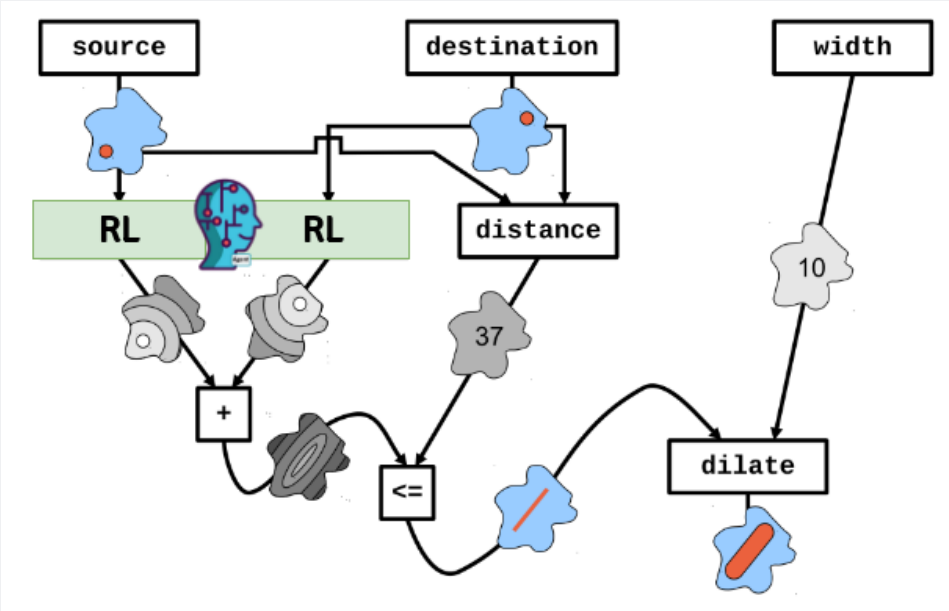


[7] G. Aguzzi, R. Casadei, and M. Viroli, "Addressing collective computations efficiency: Towards a platform-level reinforcement learning approach," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2022, Virtual, CA, USA, September 19-23, 2022*, R. Casadei, E. D. Nitto, I. Gerostathopoulos, *et al.*, Eds., IEEE, 2022, pp. 11–20. doi: 10.1109/ACSOS55765.2022.00019. [Online]. Available: https://doi.org/10.1109/ACSOS55765.2022.00019

Hybrid Aggregate computing – Algorithms

Collective Program Sketching

- *holes*: *blocks* of the aggregate program depending on *environment dynamics*
- *sketching*: *partial* specification of the aggregate program
- Idea [8]: *synthesis* of the *holes* by *learning* from the *realistic simulation*

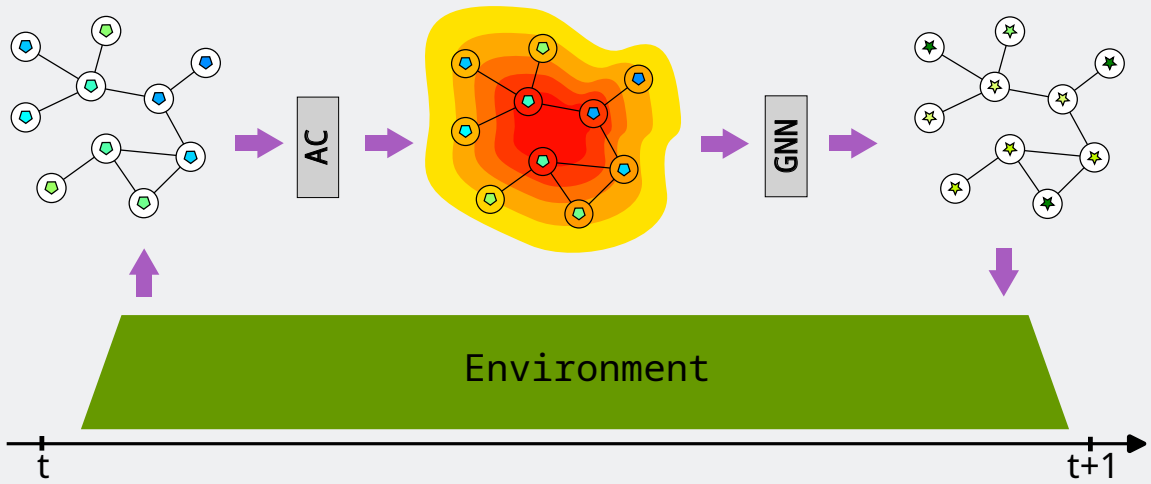


[8] G. Aguzzi, R. Casadei, and M. Viroli, "Towards reinforcement learning-based aggregate computing," in *Coordination Models and Languages - 24th IFIP WG 6.1 International Conference, COORDINATION 2022*, M. H. ter Beek and M. Sirjani, Eds., ser. Lecture Notes in Computer Science, vol. 13271, Springer, 2022, pp. 72–91. doi: 10.1007/978-3-031-08143-9_5. [Online]. Available: https://doi.org/10.1007/978-3-031-08143-9_5

Hybrid Aggregate computing – Algorithms

Field-Informed reinforcement learning

- Aggregate computing is used to *inform* the learning process
  - Speeding up the training & Reducing the search space [9]
- Field used as a way to produce a stygmergic representation of the environment
- Learning performed through *deep reinforcement learning*
- Graph-neural networks used as a policy function approximator



[9] G. Aguzzi, M. Viroli, and L. Esterle, "Field-informed reinforcement learning of collective tasks with graph neural networks," in *4th IEEE International Conference on Autonomic Computing and Self-Organizing Systems - ACSOS 2023, Toronto*, 2023

Conclusion

Takeaways

- *Swarm behaviors* are collective behaviors that emerge from the interaction of a group of agents
- *Aggregate computing* is a programming model that provides a high-level API for engineering swarm behaviors
- *ScaFi* is a Scala toolkit that provides an internal DSL, libraries, a simulation environment, and runtime support for practical aggregate computing systems development
- *MacroSwarm* is a ScaFi library for programming swarm robotics applications
- *Hybrid Aggregate Computing* is the process of combining manual and automatic design techniques to engineer swarm behaviors

Engineering Swarm Behaviours through (Hybrid) Aggregate Computing

Gianluca Aguzzi gianluca.aguzzi@unibo.it

Alma Mater Studiorum – Università di Bologna

08/04/2024

References I

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, 2013. doi: 10.1007/S11721-012-0075-2. [Online]. Available: https://doi.org/10.1007/s11721-012-0075-2.

[2] J. Beal and J. Bachrach, "Infrastructure for engineered emergence on sensor/actuator networks," *IEEE Intell. Syst.*, vol. 21, no. 2, pp. 10–19, 2006. doi: 10.1109/MIS.2006.29. [Online]. Available: https://doi.org/10.1109/MIS.2006.29.

[3] D. Pianini, M. Viroli, and J. Beal, "Protelis: Practical aggregate programming," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, R. L. Wainwright, J. M. Corchado, A. Bechini, and J. Hong, Eds., ACM, 2015, pp. 1846–1853. doi: 10.1145/2695664.2695913. [Online]. Available: https://doi.org/10.1145/2695664.2695913.

[4] G. Audrito, "FCPP: an efficient and extensible field calculus framework," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020, Washington, DC, USA, August 17-21, 2020*, IEEE, 2020, pp. 153–159. doi: 10.1109/ACSOS49614.2020.00037. [Online]. Available: https://doi.org/10.1109/ACSOS49614.2020.00037.

[5] R. Casadei, M. Viroli, G. Aguzzi, and D. Pianini, "Scafi: A scala DSL and toolkit for aggregate programming," *SoftwareX*, vol. 20, p. 101 248, 2022. doi: 10.1016/j.softx.2022.101248. [Online]. Available: https://doi.org/10.1016/j.softx.2022.101248.

[6] G. Aguzzi, R. Casadei, and M. Viroli, "Macroswarm: A field-based compositional framework for swarm programming," *CoRR*, vol. abs/2401.10969, 2024. doi: 10.48550/ARXIV.2401.10969. arXiv: 2401.10969. [Online]. Available: https://doi.org/10.48550/arXiv.2401.10969.

References II

[7] G. Aguzzi, R. Casadei, and M. Viroli, "Addressing collective computations efficiency: Towards a platform-level reinforcement learning approach," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2022, Virtual, CA, USA, September 19-23, 2022*, R. Casadei, E. D. Nitto, I. Gerostathopoulos, *et al.*, Eds., IEEE, 2022, pp. 11–20. doi: 10.1109/ACSOS55765.2022.00019. [Online]. Available: https://doi.org/10.1109/ACSOS55765.2022.00019.

[8] G. Aguzzi, R. Casadei, and M. Viroli, "Towards reinforcement learning-based aggregate computing," in *Coordination Models and Languages - 24th IFIP WG 6.1 International Conference, COORDINATION 2022*, M. H. ter Beek and M. Sirjani, Eds., ser. Lecture Notes in Computer Science, vol. 13271, Springer, 2022, pp. 72–91. doi: 10.1007/978-3-031-08143-9_5. [Online]. Available: https://doi.org/10.1007/978-3-031-08143-9_5.

[9] G. Aguzzi, M. Viroli, and L. Esterle, "Field-informed reinforcement learning of collective tasks with graph neural networks," in *4th IEEE International Conference on Autonomic Computing and Self-Organizing Systems - ACSOS 2023, Toronto*, 2023.