Co-Fields & TOTA

field-based coordination models

Lucio Tudisco

Sistemi Multi-Agente LS

16 Marzo 2009

Outline

The challenge of coordination in MAS

- Coordination - *the glue that binds separate activities into an ensemble* [1] - is a key issue in concurrent and distributed systems.
- Coordination (meta) models, in particular, are essential to properly manage the interactions in MAS.
- A (meta) model of coordination suitable for *agent oriented software engineering* must meet the following minimum requirements;
  - Openness;
  - Complexity;
  - Adaptativity;
  - Distribution;
  - Technological etherogeneity;
  - ...

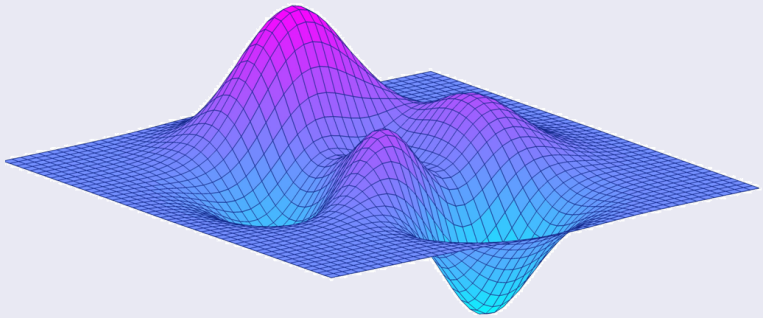
The Co-Field coordination model

Definition of Co-Fields [3]

Co-Fields are simple data structures spread across an environment according to a field-specific propagation rule.

A Co-Field is characterized by:

- An unique identifier;
- A location-dependent numeric value;
- An application-dependent propagation rule;



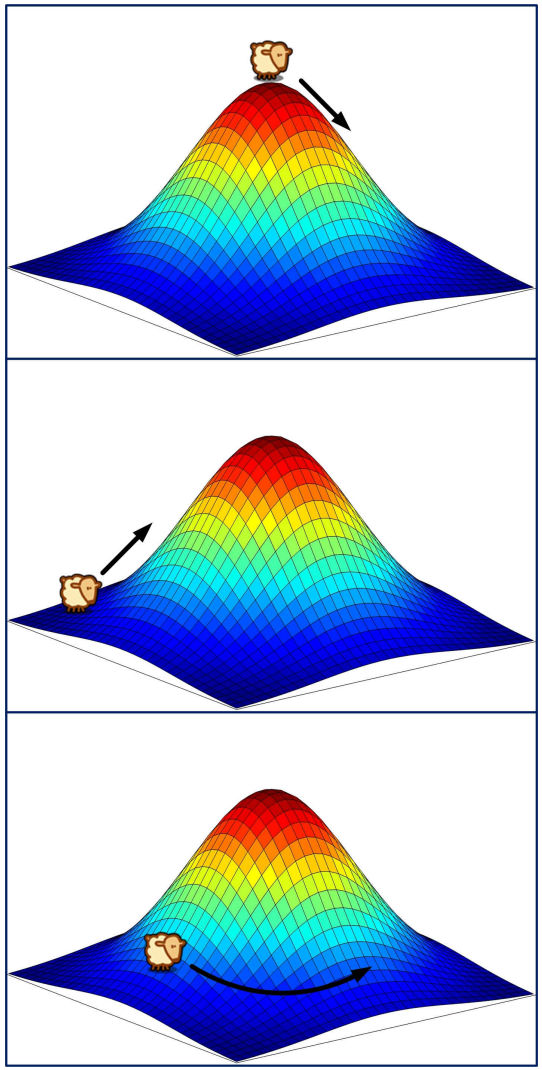
Schema of the Co-Fields model [3]

1. Environment is abstracted by fields.
2. Agents move following the waveform of these fields.
3. Environmental dynamics and agents' movements induce changes in the fields surface, composing a feedback cycle that influences agents' movement.
4. This feedback lets emerge forms of coordination.

Analytical description of agents' coordination

- In the Co-Fields model agents compute a linear combination of the fields they perceive. The result of this coordination is itself a field, called agent's *coordination field*.
- Agents typically move following downhill (uphill) the decrease (increase) of their coordination field.
- Let i be a generic agent, $(x_1^i(t),x_2^i(t),...,x_n^i(t))$ its coordinates in a particular space and $CoorField_i(X_1, X_2, ..., X_n)$ its *coordination field*.
- The behavior of the agent i is governed by the following system of PDE:

$$\frac{dx_j^i}{dt} = \pm v \frac{\partial CF_i(X_1, X_2, ..., X_n, t)}{\partial X_j} \qquad j = 1, 2, ..., n$$



Case study: a pack of predators surrounds a prey

- In nature, predators hunt for the prey maintaining a suitable distance from other predators.
- How is it possible to model this behavior with Co-Fields?
- The prey, located at coordinates (X_m, Y_m), generates the following field $(k_m, h_m > 0)$:

$$Field_{prey}(x, y, t) = -k_m e^{-h_m((x-X_m)^2+(y-Y_m)^2)}$$

- The predator i, located at coordinates (X_{wi}, Y_{wi}), generates the following field $(k_w, h_w > 0)$:

$$Field_{predator_i}(x, y, t) = k_w e^{-h_w((x-X_{wi})^2+(y-Y_{wi})^2)}$$

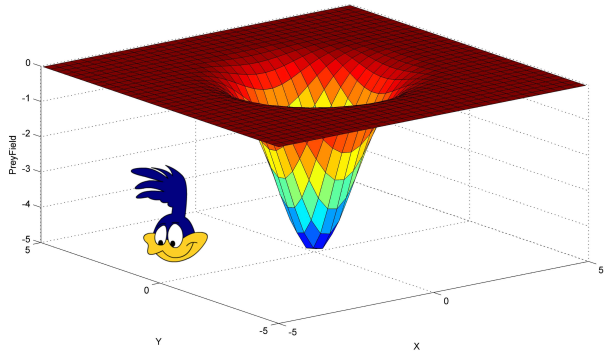


Figure: $Field_{prey}$

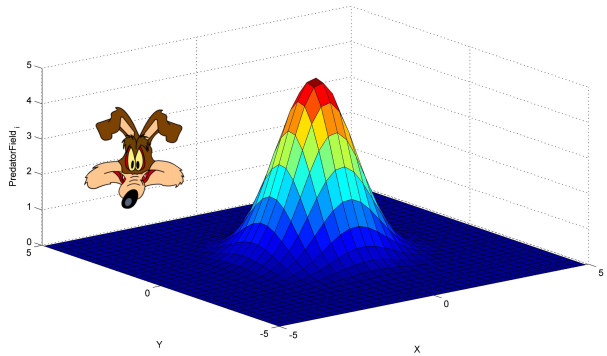


Figure: $Field_{predator_i}$

Case study: a pack of predators surrounds a prey (cnt.)

- The fields generated by prey and predators are updated in real-time depending on their movements.

- The prey runs away from predators following the decrease of its coordination field, consisting in the linear combination of all the predators' fields.

$$CF_{prey}(x, y, t) = \sum_{i=0}^{n} Field_{predator_i}$$

- Each predator sorrounds the prey following the decrease of its coordination field, consisting in the linear combination between the prey's field and all the other predators' fields.

$$CF_{predator_i}(x, y, t) = Field_{prey} + \sum_{\substack{j=0 \\ j \neq i}}^{n} Field_{predator_i}$$

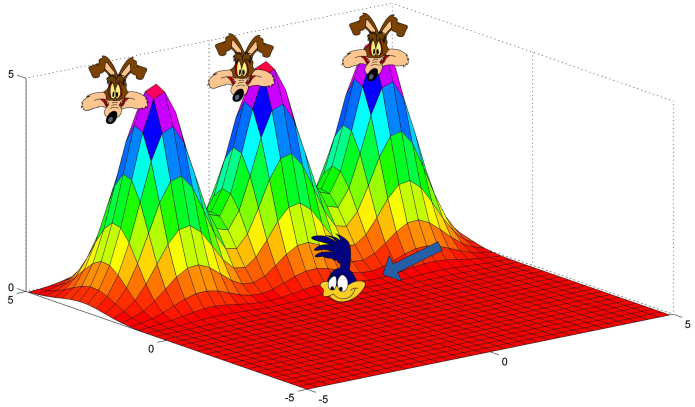


Figure: CF_{prey}

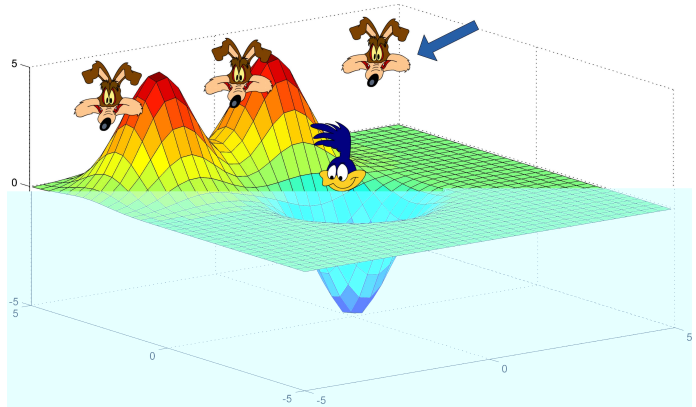


Figure: $CF_{predator_i}$

Case study: a pack of predators surrounds a prey (cnt.)

- The PDE system that governs the prey escape and the predators surrounding strategy is the following:

$$\begin{cases} \dfrac{dx_m}{dt} = \dfrac{\partial CF_{prey}}{\partial x_m} \\ \dfrac{dy_m}{dt} = \dfrac{\partial CF_{prey}}{\partial y_m} \\ \dfrac{dx_{w_i}}{dt} = \dfrac{\partial CF_{predator_i}}{\partial x_{w_i}} \quad i = 1,2,3 \\ \dfrac{dy_{w_i}}{dt} = \dfrac{\partial CF_{predator_i}}{\partial y_{w_i}} \quad i = 1,2,3 \end{cases}$$

- The result of a numerical integration of this system in the case of one prey and three predators is displayed in the figure to the right.

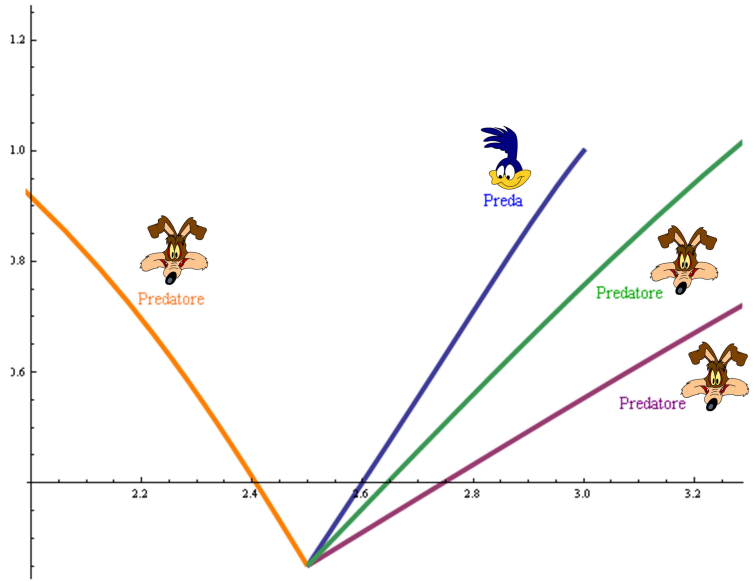


Figure: Trajectories of prey and predators. Predators, in particular, reach the prey from three different directions as expected in a coordinated surrounding strategy.

TOTA: Tuples On The Air

- TOTA is a middleware infrastructure concived as a support for the Co-Fields coordination model.
- TOTA relies on spatially distributed tuples to be injected in a peer-to-peer network of possibly mobile nodes, each running a local version of the TOTA middleware.
- A TOTA tuple is defined as a triple *T=(C,P,M)* where:
  - *C* is a *content*
  - *P* is a *propagation rule*
  - *M* is a *maintenance rule*
- The TOTA middleware is active and adaptive. It maintains, in fact, tuples propagation coherent despite network dynamism.

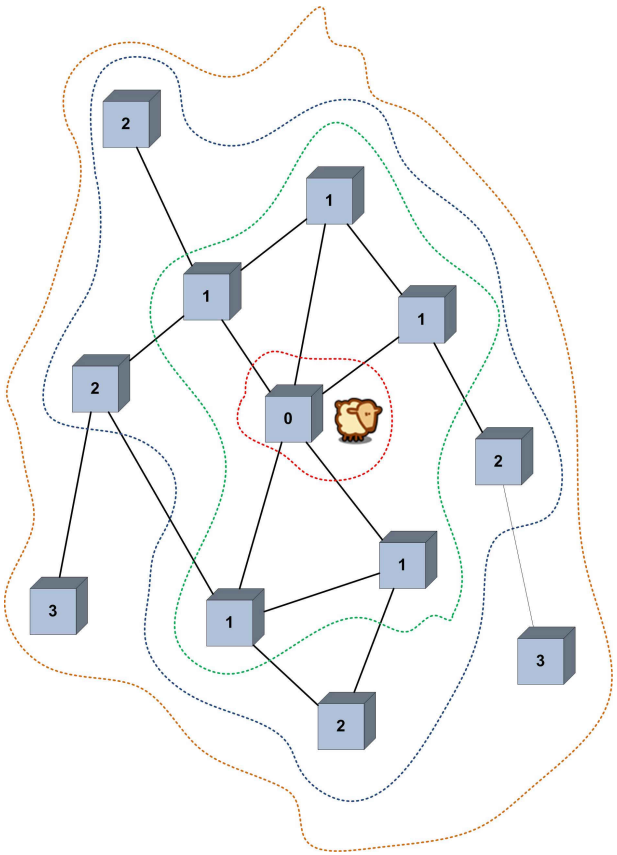


Figure: TOTA Network

TOTA Architecture

- TOTA is entirely developed in Java, thus it can run on any JVM-equipped computer device.
- The TOTA architecture is composed of three main parts:
  1. The TOTA API: it is the interface between the application and the middleware.
  2. The EVENT INTERFACE: it is the component responsible for notifying the application about the income of a new tuple or about the connection (disconnection) of a new node to the node's neighborhood.
  3. The TOTA ENGINE: it is the component responsible for storing tuples, applying their propagation and maintenance rules, managing the references to neighboring nodes and monitoring network reconfiguration and the income of new tuples.

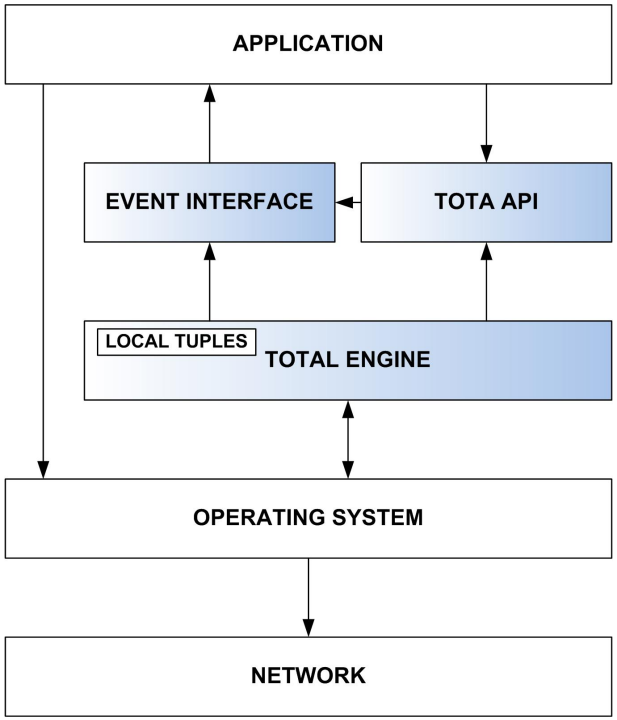


Figure: TOTA Architecture

TOTA API ... in a single slide

```
public void inject(TotaTuple tuple);

public Vector read(Tuple template);

public Vector readOneHop (Tuple template);

public Tuple keyrd (Tuple template);

public Vector keyrdOneHop(Tuple template);

public Vector delete(Tuple template);

public void subscribe(Tuple template, ReactiveComponent comp, String rct);

public void unsubscribe(Tuple template, ReactiveComponent comp);
```

Specifying Tuples

- TOTA tuples share all the following abstract structure:

```
public abstract class TotaTuple {

  protected TotaInterface tota;

  public void init(TotaInterface tota) {
     this.tota = tota;
  }

  public abstract void propagate();

  public void react(String reaction, String event);

}
```

- To simplify the tuples specification, however, TOTA provides a library of class hierarchies from which the programmer can inherit to create specific tuples abstracting from low-level details.

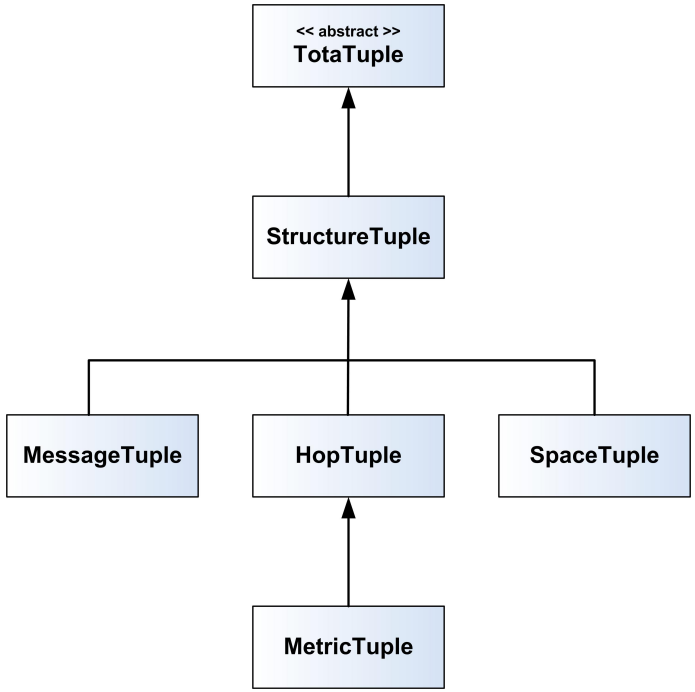


Figure: TOTA tuples class hierarchy

Case study: a group of agents coordinate themselves to move in formation

- In nature, birds form flocks following this simple swarm alghorythm: each birds maintains a specified distance from the nearest birds and matches their speed.
- How is it possible to model this behavior with Co-Fields?
- Each agent i generates a $Flock_i$ field described by the equation:

$$Flock_i(x, y, t) = d^4 - 2a^2d^2$$

$$d = \sqrt{(x - X_i)^2 + (y - Y_i)^2}$$

- The resulting coordination field is a regular grid. Agents move in the local minima of this field.

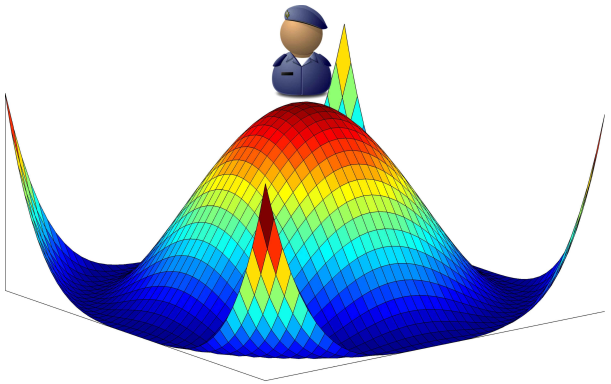


Figure: Field $Flock_i$

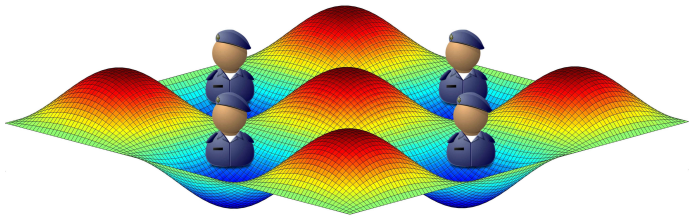


Figure: Coordination Field

Implement flocking behavior with TOTA

The tuple `FlockingTuple` propagates in the TOTA network a field assuming its minima in nodes far `RANGE` hops from the source.

```
public class FlockingTuple extends HopTuple {

  private int RANGE, value;

  public FlockingTuple(int RANGE) {
    this.RANGE = RANGE;
    value = RANGE;
  }

  protected void changeTupleContent() {
     super.changeTupleContent();
     value = (hop <= RANGE)?value-1: value+1;
  }

}
```

Agent inject flocking tuples then they follow flocking tuples downhill.

```
public class FlockingAgent implements AgentInterface {

   private TotaMiddleware tota;

   public void run() {

     FlockingTuple ft = new FockingTuple ();
      ft.setContent(peer.toString());
      tota.inject(ft);

      while(true) {
        FlockingTuple query = new FlockingTuple();
        Vector v = tota.read(query);
        GenPoint destination = getDestination(v);
        this.move(destination);
      }

   }

}
```

Demo

(Loading...)

References


D. Gelernter e N. Carriero.,
"Coordination languages and their significance",
Communications of the ACM, Volume 35 , Issue 2, pp. 97-107, 1992.


Sito ufficiale dell'*Agents and Pervasive Computing Group*,
http://agentgroup.ing.unimo.it


M. Mamei e F. Zambonelli,
"Field-based Coordination for Pervasive Multiagent Systems",
Springer, 2005.


M. Mamei, L. Leonardi e F. Zambonelli
"Co-Fields: A Unifying Approach to Swarm Intelligence",
3rd International Workshop on Engineering Societies in the Agents World (ESAW), LNAI 2577, Springer Verlag, pp. 68-81, 2003.


M. Mamei, F. Zambonelli e L. Leonardi,
"Distributed Motion Coordination with Co-Fields: A Case Study in Urban Traffic Management",
6th IEEE Symposium on Autonomous Decentralized Systems (ISADS 2003), IEEE CS Press, Pisa (I), 2003.

References

M. Mamei e F. Zambonelli,
"Programming Pervasive and Mobile Computing Applications with the TOTA Middleware",
2nd IEEE International Conference on Pervasive Computing and Communication, IEEE CS Press, Orlando (FL), 2004.

M. Mamei, F. Zambonelli e L. Leonardi,
"Co-Fields: A Physically Inspired Approach to Distributed Motion Coordination",
IEEE Pervasive Computing, Volume 3 , Issue 2, pp. 52-61, 2004.

M. Mamei e F. Zambonelli,
"Programming Stigmergic Coordination with the TOTA Middleware",
International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press, Utrecht, NL, 2005.