

A Catalogue of Decentralised Coordination Mechanisms for Designing Self-Organising Emergent Applications

Tom De Wolf and Tom Holvoet

Report CW458, August 2006



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Catalogue of Decentralised Coordination Mechanisms for Designing Self-Organising Emergent Applications

Tom De Wolf and Tom Holvoet

Report CW 458, August 2006

Department of Computer Science, K.U.Leuven

Abstract

There is little or no guidance to systematically design a self-organising emergent solution that achieves the desired macroscopic behaviour. This paper describes decentralised coordination mechanisms such as digital pheromones as design patterns, similar to patterns used in mainstream software engineering. As a consequence, a structured consolidation of best practice in using each coordination mechanism becomes available to guide engineers in applying them, and to directly decide which mechanisms are promising to solve a certain problem. As such, self-organising emergent solutions can be engineered more systematically, which is illustrated in a packet delivery service application. This document includes extensive pattern descriptions for digital pheromones, gradient fields, market-based coordination, tag-based coordination, and token-based coordination.

Keywords : design patterns, emergence, decentralised coordination.

ACM Classification : Primary : I.2.11 Distributed Artificial Intelligence, Secondary : D.2.11. Software Architectures

A Catalogue of Decentralised Coordination Mechanisms for Designing Self-Organising Emergent Applications.

Tom De Wolf and Tom Holvoet

AgentWise@DistriNet Research Group
Department of Computer Science, KULeuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{Tom.DeWolf, Tom.Holvoet}@cs.kuleuven.be
<http://www.cs.kuleuven.be/~tomdw>

Abstract There is little or no guidance to systematically design a self-organising emergent solution that achieves the desired macroscopic behaviour. This paper describes decentralised coordination mechanisms such as digital pheromones as design patterns, similar to patterns used in mainstream software engineering. As a consequence, a structured consolidation of best practice in using each coordination mechanism becomes available to guide engineers in applying them, and to directly decide which mechanisms are promising to solve a certain problem. As such, self-organising emergent solutions can be engineered more systematically, which is illustrated in a packet delivery service application. This document includes extensive pattern descriptions for digital pheromones, gradient fields, market-based coordination, tag-based coordination, and token-based coordination.

1 Introduction

Modern distributed systems exhibit an increasingly interwoven and completely decentralised structure [1] (e.g. ad-hoc networks, transportation systems, etc.). Different sub-systems interact with each other in many, often very complex, dynamic, and unpredictable ways. More and more systems need to achieve their requirements autonomously [2]. A promising approach is to use a group of agents that cooperate to autonomously achieve the required system-wide or macroscopic behaviour using only local interactions, local activities of the agents, and locally obtained information. Such decentralised multi-agent systems (MASs) exhibit *self-organising emergent behaviour* [3].

When engineering a self-organising emergent solution, the problem-solving power mainly resides in the interactions and coordination between the agents instead of in the intelligent reasoning of single agents. Therefore, a major architectural design decision is the choice of suitable decentralised coordination mechanisms such as digital pheromones [4], gradient fields [5], market-based control [6], tags [7], or tokens [8]. Many of such mechanisms are already applied to a number of case studies in literature [4,9,5,10,11,12,13,7,8,6,14,15,16,17,18,19]. However, a fundamental problem is the lack of guidance on how to systematically choose and use the most suitable coordination mechanism. The main reason is that, currently, all existing knowledge and

best practice on coordination mechanisms is spread over hundreds of papers without a clearly structured and directly usable description of the mechanisms.

The main contribution of this paper is twofold. First, the paper shows how decentralised coordination mechanisms can be structurally described as design patterns. Secondly, the paper illustrates (in a packet delivery application) how an engineer can use these patterns to systematically choose how to coordinate agents and achieve the desired global behaviour. Section 2 motivates design patterns as a description to support engineers in their choice of decentralised coordination mechanisms. The section also outlines in detail the structure of the pattern description used in the rest of the paper. After that, sections 3, 4 through 8 give an usable pattern summary and detailed pattern description of a number of coordination mechanisms. In section 9, a case study on a packet delivery service illustrates how design patterns allow to engineer more systematically a self-organising emergent solution. Finally, section 10 concludes and discusses future work.

2 Decentralised Coordination Mechanisms as Design Patterns

Typically, engineering MASs means having 99% of the effort go to conventional computer science and only 1% involves the actual agent paradigm [20]. MAS should be allocated a correct role within mainstream software engineering, rather than positioning MASs as a radically new approach [21]. Therefore, to engineer self-organising emergent MASs, developers should exploit conventional software technologies and techniques wherever possible [21]. Such exploitation speeds up development, avoids reinventing the wheel, and enables sufficient time to be devoted to the value added by the multi-agent paradigm [22]. From this point of view, [23] proposes a step plan based on an existing industry-ready software engineering process, i.e. the Unified Process [24]. The UP process is customised to explicitly focus on how to address the desired macroscopic behaviour of self-organising emergent MASs. Figure 1 shows that almost every discipline in the UP process is customised.

This paper focusses on the Design which emphasises a solution that fulfills the requirements, rather than its implementation. More specifically the focus is on the coarse-grained architectural design. The author of [25] states that architectural design is partially a science and partially an art. The *science* of architecture is the collection and organisation of information about architectural significant requirements (e.g. non-functional and macroscopic functionality). The *art* or architecture is making skillful choices and constructing a solution that meets the requirements, taking into account trade-offs, interdependencies, and priorities. The ‘art’ of architectural design is the creative step where designers exploit knowledge and best practice from a number of areas (e.g. architectural styles and patterns, technologies, pitfalls, and trends) as a guide to reach a suitable solution. For self-organising emergent systems the main source of knowledge to exploit in this creative step are the different mechanisms to coordinate the desired macroscopic functionality such as digital pheromones [4], gradient fields [5], and market-based coordination [6]. This paper captures this knowledge on decentralised coordination mechanisms as architectural design patterns. In what follows, first

a motivation is given for using a design pattern format. After that, an outline of the structure used in this paper to describe the patterns is given and motivated.

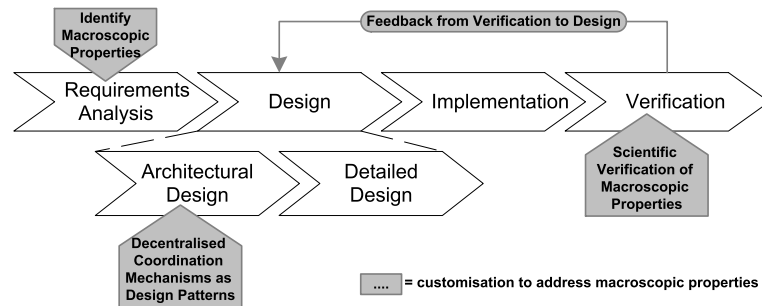


Figure 1. A Unified Process engineering iteration annotated with customisations for issues specific for self-organising emergent MASs.

2.1 Motivation for Design Patterns

As mentioned earlier, the main problem-solving power resides in the coordination between agents and a major architectural design decision concerns the choice of one or more decentralised coordination mechanisms that are suitable to achieve the desired macroscopic behaviour. The mechanism used to coordinate has a strong impact on what is and can be communicated [9]. A lot of such mechanisms are used in literature but experience and knowledge about how to use them is currently spread out over hundreds of articles. A more structured and useful description is needed to consolidate this best practice.

In mainstream software engineering, current best practice and knowledge about known solutions is captured in what is called ‘design patterns’. The most famous reference is the Gang of Four design patterns book [26]. This paper supports using decentralised coordination mechanisms by describing them as design patterns, similar to patterns used in mainstream software engineering. An important issue in engineering self-organising emergent solutions is to understand for which kind of macroscopic behaviour each coordination mechanisms is useful, which is more appropriate in which situation, etc. Design patterns describe each decentralised coordination mechanism in a structured way, inherently including this kind information. Using such a set of structured patterns, self-organising emergent system can be designed more systematically.

2.2 The Pattern Description Format

Patterns can be described at several levels of abstraction. The patterns in the Gang of Four book [26] are described at the class or implementation level. Another level of abstraction to describe design patterns is the architectural or even conceptual level in

which the focus is more on the coarse-grained and conceptual structure. The decentralised coordination mechanisms in this paper are described at the architectural and conceptual level. The class level is not described because such mechanisms can be implemented in multiple ways and little is known about the best way to do this.

This paper uses the guidelines and patterns for describing patterns from [27]. As such, the decentralised coordination mechanisms are described in a format known in mainstream software engineering which promotes their usage. For example, also the Gang of Four patterns book [26] uses a similar format. The general structure is extended in subsections to better describe issues specific for decentralised coordination in self-organising emergent solutions. The format used involves the following sections:

- **Pattern Name/Also Known As:** A clear name and aliases referring to the solution used or a useful metaphor. The name is given in the title of the pattern's section.
- **Context/Applicability:** The circumstances in which the problem being solves requires a solution. Often described via a 'situation'. In this paper this context typically indicates when a self-organising emergent solution is promising.
- **Problem/Intent:** What is solved by this pattern? Engineers compare this section with their problem in order to select coordination mechanisms.
- **Forces:** Often contradictory considerations that must be taken into account when choosing a solution to a problem.
- **Solution:** A description of how the problem is solved and described in close relation to the forces it resolves. This section has a more detailed structure:
 - *Inspiration:* Most coordination mechanism are inspired by some natural, biological, physical, or social phenomena.
 - *Conceptual Description:* A conceptual description of how the inspirational mechanism works and is used in computer systems. This section allows to map the concepts used in the coordination mechanism to domain-specific entities in the system under construction, i.e. the concepts and their relationships indicate what is needed in order to use this coordination mechanism.
 - *Parameter Tuning:* Typically, such coordination mechanisms have a lot of parameters that need to be tuned. This section enumerates them and gives some guidelines to tune them.
 - *Infrastructure:* Some mechanisms need a supporting infrastructure. More specifically, what is needed to support the design at the class level.
 - *Characteristics:* Using the mechanism imposes some characteristics on the solution including advantages, disadvantages, and other useful properties.
- **Related Mechanisms/Patterns:** An enumeration of patterns that are related in which the differences and similarities are emphasised.
 - *Variations:* Variations of the same pattern which can be more general or (domain) specific variations.
 - *Other Coordination Mechanisms:* Alternative coordination mechanisms that solve the same (or related) problem.
- **Examples/Known Uses:** Examples of known uses of the pattern in case studies.

In addition to a separate description for each pattern, a 'Problem/Solution Summary' is provided to help the reader find the pattern(s) which solve(s) their specific problems. Such a summary is typically a table with a brief description of the problem

each pattern solves and how. In what follows, section 3 gives such a summary for a limited set of widely used coordination mechanisms and sections 4 through 8 describe these patterns in more detail following the structure given in this section.

3 Problem/Solution Summary

This section summarises the patterns described in this paper by giving a so called 'Problem/Solution Summary'. An engineer can use this to find the pattern(s) or coordination mechanism(s) which solve(s) their specific problem(s). The Problem/Solution Summary can be found in Table 1 where a brief description of the problem and the corresponding solution is linked to the pattern to use.

Table 1. Problem/Solution Summary

Problem(s)	Solution	Pattern
Spatial Source to Destination Routing, Task Recruitment, Relation Identification, Integration of various information sources	<i>Agents explicitly search</i> for goals, tasks, or related items and drop pheromones to gradually form historical paths for other agents to follow to the goal. Reinforcement of an existing path by other agents can be seen as a reinforcement of the relation between source and destination. Evaporation, Aggregation, and Propagation keep the pheromones up-to-date and support integration of various information sources.	Digital Pheromone Paths [4]
Spatial Movement, Pattern Formation, Structure Formation, Routing, Integration of Contextual Information	Spatial, contextual, and coordination-related information is automatically and instantaneously spread/propagated by the environment as multiple computational fields. <i>Agents simply follow the “waveform”</i> of these fields to coordinate, i.e. <i>no explicit exploration</i> . The spatial information comes to the agents instead of agents explicitly searching.	Gradient Fields [5]
Resource Allocation in general (resource=task, power, bandwidth, space, time, etc.) , Integration of resource Usage/Need Information	A virtual market where resource users sell and buy resource usage with virtual currency. The price evolves according to the market dynamics and indicates a high (high price) or low (low price) demand. This information is used by agents to decide on using the resource or not. Economic market theory states that the prices converge to a stable equilibrium.	Market-based Coordination [6]
Trust and reputation, Team-formation , Discourage selfish behaviour in Teams, Specialisation of skills within Teams	Agents put and modify tags on other agents and a team is formed by only collaborating with agents with the same tag or some other condition. If tags indicate how well agents behaved in collaborations with others then trust and reputation information can be available.	Tags [7]
Resource Access Control/Allocation, Role Allocation, Enforce Organisation Structure, Information Sharing	Information, resources, or roles are represented by a token. Only the holder has exclusive access to the information and resource. Holding a role token commits to executing that role. Tokens are passed among agents to get adaptive coordination.	Tokens [8]
etc.	etc.	etc.

4 Pattern 1: Digital Pheromone Paths

Also Known As: Pheromone Coordination, Digital Pheromones, Pheromone Foraging, Pheromones.

4.1 Context/Applicability

You are building a solution for an application domain that requires that multiple autonomous entities coordinate in a decentralised way to achieve a common and globally coherent goal. The coordination mechanisms also has to be robust and flexible in the face of frequent changes.

More specifically, the autonomous entities are situated in an environment (physical or logical) which can be extended with needed infrastructure and the environment structure represents the problem to be solved. Some kind of spatial movement of the autonomous entities is required or information about the spatial location of some entities has to be exchanged.

Local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. For example in network routing [28], information propagation delays, and the difficulty to model the whole network dynamics under arbitrary traffic patterns, make the general routing problem intrinsically distributed. Routing decisions can only be made on the basis of local and approximate information about the current and the future network states.

4.2 Problem/Intent

- *Spatial Source to Destination Routing:* How to guide autonomous entities or agents to move or route themselves adaptively and as optimal as possible through the environment structure, i.e. the problem structure? This has to be achieved *in the face of changes* in a dynamic environment (e.g. obstacles, failure, new goal locations, etc). In other words, how to *spread up-to-date information* to agents *on spatial locations, direction-to locations, and other information* about that location or locations in between, to decide on their next move. Possibly to *attract agents to certain locations* or in a certain direction.
- *Integration of various information sources:*
 - How to process information in a completely distributed and decentralised environment [29]? The information should be stored close to where the information that is integrated is generated, and close to where it will be used.
 - How to integrate diverse information sources in one coordination mechanism [29]? Various types of information from various sources should be possible.
- *Task Recruitment:* When moving to the locations implies execution of a certain task then the problem to solve becomes how to recruit other agents for tasks [30]?
- *Relation Identification:* The problem to solve becomes how to identify relationships between different items when viewing the formation of routes and paths between locations as the construction of a relationship between those locations [31].
- How to make the *agents themselves responsible to construct and spread the information* needed to coordinate?

- How to achieve coordination in a way that *agents can leave and join* the coordination process at any time and any location?
- How to have an “*optimised*” coordination (*not necessarily optimal*) without losing *robustness and adaptiveness* [32]?

4.3 Forces

- Explore vs. Exploit: In order to be adaptive the solution has to explore sufficiently compared to only exploiting already known information. Otherwise the approach can get trapped in so called local optima or never find new targets at all. However, too much exploration may result in an approach that is very inefficient.
- Centralised vs. Decentralised: A decentralised solution such as this pattern often requires a huge amount of communication and coordination overhead which a centralised solution has not. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context. A centralised solution also has a huge amount of communication overhead if the information needed to control the system is intrinsically distributed and a lot of it has to be aggregated to the central node (e.g. large-scale heavily loaded (network) systems). Often, a model of the complete state of a system cannot be obtained at all. As such, decentralised control is the only alternative.
- Optimality vs. Robustness/Flexibility: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. However, an optimal solution only exists with respect to a static situation, which is never reached in the face of frequent changes. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible.
- *Responsibility of Environment vs. Agents*: Coordination often needs complex processing and communication. There is a trade-off to make the agents themselves or the environment in which they are situated responsible for this processing and communication. Making the agents responsible allows agents to explicitly reason about and control how information is distributed but sometimes requires complex algorithms to do that reasoning. On the other hand, making the environment responsible allows agents to simply be guided by the results in the environment but the agents are no longer in control of the information distribution.

4.4 Solution

Inspiration. Inspired by nature, many self-organising emergent systems communicate through “indirect communication” or “stigmergy” [30,28,32]. Stigmergy is a method of communication in which the individual entities of the system communicate information with each other by modifying their local environment. Stigmergy was first observed in nature - ants coordinate their food foraging by constructing and following networks of pheromone paths that connect their nests with available food sources. Highly volatile chemical substances, called pheromones, provide the means for indirect communication as marks in the environment with information to assist other ants at a later time. When an ant finds food it heads back to the nest leaving a pheromone trail, a trail of scent. At their current location, ants observe the local strength and the local gradient of the

perceived pheromone strengths and gradients are incorporated into the probabilistic decision processes of the single agent. The agent can then follow a pheromone path. As such information is exchanged in a feedback cycle between agents on which they can coordinate.

In more detail, each agent needs to reach a certain *destination* (e.g. food source) by following pheromone paths and to execute an action at that destination (e.g. pick-up food). An agent has a probabilistic or stochastic behaviour which can be accomplished by the following set of simple rules [4,32]:

- Rule 1: Move randomly or following another probabilistic exploration strategy.
- Rule 2: If you find yourself at the *destination* and you have not just done your *destination action* there (e.g. not holding food which you just picked up), then do your *destination action* (e.g. pick-up food).
- Rule 3: If you find yourself at the *source* where you originate from (e.g. nest) and you are returning from the destination (e.g. carrying food), do your *source action* (e.g. drop the food).
- Rule 4: Select every step from the current distribution over possible directions. If no pheromones are sensed, perform a randomised search for the destination, executing Brownian motion. If pheromones are sensed, bias the probabilistic selection in favour of the scent's direction, i.e. strongest pheromone has preference.
- Rule 5: If you are returning from the destination to the source location (e.g. holding food), drop fixed amounts of pheromone at a constant rate as you walk.

A pheromones is not static data put into the environment but are managed by that environment by the execution of three primary *pheromone actions*, inspired by the dynamics of biological pheromones [4,30,34,32]:

- *Evaporate over time*: The continuous (at a certain *evaporation rate*) decreasing of the local pheromone strength is governed by a dispersion function of the form $s(t + 1) = s(t) * E$, where E is a constant *evaporation factor* between one and zero. This serves to forget old information that is not refreshed or reinforced, e.g. a pheromone trail to an exhausted food source is no longer refreshed by other ants. As such, pheromones support truth maintenance of information.
- *Aggregate at same locations*: Pheromones can be deposited and withdrawn from a location. Deposits are added to the current amount of pheromones located at that place. As such, information fusion and aggregation is supported.
- *Propagate in space*: Pheromones propagate, at a certain *propagation rate*, from a location in space to its neighbouring locations. The amount that is propagated is determined by the *propagation factor*. The act of propagation causes diffused pheromone to be formed. As such, information diffusion and spreading is supported.

Note that an engineer can customise every pheromone action by specifying the wanted *propagation rule*, *evaporation rule*, and/or *aggregation rule*. These rules determine how and under which conditions a pheromone is propagated, evaporated, and aggregated. For example, in [34] a formal model is given of the pheromone dynamics that uses a specific evaporation and propagation rule. Note that this description already makes a distinction between multiple types or flavours of pheromones (see

variations in section 4.5). Each location maintains a scalar variable corresponding to each pheromone flavour. It performs the basic functions of aggregation, evaporation, and propagation. The underlying mathematics of the field developed by such a network of places, including critical stability theorems, rest on two fundamental equations. The parameters governing the pheromones are:

- $P = [p_j]$ = set of place agents
- $N : P \rightarrow P$ = neighbour relation between place agents. Thus the place agents form an asymmetric multigraph
- $s(\Phi_f, p, t)$ = strength of pheromone flavour f at place agent p and time t
- $d(\Phi_f, p, t)$ = external deposits of pheromone flavour f within the interval $(t - 1, t]$ at place agent p
- $g(\Phi_f, p, t)$ = propagated input of pheromone flavour f at time t to place agent p
- $E_f \in (0, 1)$ = evaporation factor for flavour f
- $G_f \in [0, 1)$ = propagation factor for flavour f
- T_f = threshold below which $s(\Phi_f, p, t)$ is set to zero

The first equation describes the evolution of the strength of a single pheromone flavour at a given location:

$$s(\Phi_f, p, t + 1) = E_f * [(1 - G_f) * (s(\Phi_f, p, t) + d(\Phi_f, p, t)) + g(\Phi_f, p, t)] \quad (1)$$

E_f reflects the evaporation of the pheromone, the $1 - G_f$ factor calculates the amount remaining after propagation to its neighbours, $s(\Phi_f, p, t)$ represents the amount of pheromone from the previous cycle, $d(\Phi_f, p, t)$ represents the total deposits made since the last update cycle, and $g(\Phi_f, p, t)$ represents the total pheromone propagated in from all the neighbours of p . Each location applies this equation to each pheromone flavour once during every update cycle.

The second fundamental equation described the propagation received from the neighbouring locations:

$$g(\Phi_f, p, t) = \sum_{p' \in N(p)} \frac{G_f}{|N(p')|} (s(\Phi_f, p', t - 1) + d(\Phi_f, p', t - 1)) \quad (2)$$

This equation states that each neighbour location p' propagates a portion of its pheromone to p each update cycle, the portion depending on the parameter G_f and the total number of its neighbours.

Parameter Tuning. Each pheromone has a number of parameters, or “settings”, that tune the pheromones to the task [34,33,4,28]:

- *Evaporation factor:* the fraction of the pheromone that remains after the evaporation cycle. The evaporation rate impacts the amount of exploration versus the amount of exploitation. If the evaporation rate is high, information about previously discovered solutions is forgotten more rapidly and the algorithm will explore more, and vice versa. If the pheromone does not evaporate fast enough, too many agents might be attracted into the wrong direction. Evaporation of pheromones guarantees

that a change in the problem structure eventually becomes known at the relevant locations in a change of the local pheromone field. Proper tuning of the evaporation rate has an impact on the performance and convergence of the *Digital Pheromone Paths* coordination mechanism.

- *Propagation factor*: the fraction of the pheromone in a location that is distributed equally among all the neighbours. This is a value between zero and 1. A factor of zero prevents all propagation.
- *Update cycle times*: the regular time at which updates are done:
 - Propagation Rate: rate with which pheromones propagate
 - Evaporation Rate: rate with which pheromones evaporate
 - Refresh Rate: rate with which deposits are made by agents
- *Minimal strength* of a pheromone: if after propagation and evaporation, the amount of pheromone in a location falls below this strength level then the pheromone strength is set to zero, i.e. the pheromone is removed
- *Initial strength* of a pheromone: when a pheromone is created and deposited in the environment it has an initial value for its strength.

When tuning these parameters, the dynamics of the changes (frequency, impact, etc.) and the constraints of the problem structure have to be taking into account [4]. A formal model and theory such as described above and in [4] can serve as a means to tune these parameters.

Infrastructure. To be able to use *Digital Pheromone Paths* the environment needs to support storing pheromones at locations and applying the pheromone dynamics to them (evaporation, aggregation, propagation). This application-independent software component of the possibly distributed runtime environment is called the Pheromone Infrastructure in the environment [4,33]. Each location in the environment provides the *required services* in the pheromone infrastructure:

- Agents can access the local data, i.e. observing pheromones and their local strength on the neighbouring locations.
- Agents can modify the local data, i.e. depositing pheromones.
- Agents can directly communicate with agents at the same location and can acquire topological information (i.e. what are the neighbours).
- The local pheromone dynamics is automatically enforced: aggregation according to the aggregation rule, evaporation according to the evaporation factor and rule, and propagation according to the propagation factor and rule.

Characteristics.

- The structure of the environment in which the agents are working should reflect the current “problem” the ants are working on and the pheromone concentration and distribution should reflect the current “solution” to that problem [32].
- Mathematically, the pheromone networks form minimum spanning trees [4,29]. Thus, they minimise the path length ants or agents travel, i.e. the *shortest path* is

found using a stochastic decision policy based only on local information represented by the pheromone trail deposited by other ants [28]. If multiple trails are found then the shortest route will dominate because the shortcut will result in more trips per agent per unit of time and thus the scent on that trails will be stronger and draw more agents.

- The spreading of global information (trails to food sources) and the feedback on the behaviour of the ant society (ants are attracted by scents) guides the system towards a solution, which is *optimised but not necessarily optimal*. The exploration of ants (random walk) guarantees that the society shows *robust behaviour*. As a result the society as an acting entity will find an “*optimised*” *solution without losing robustness and adaptiveness* [32].
- *Diversity of Information* [29]: Various types of information from various sources are possible. Multiple flavors of pheromones (see variations under ‘Related Mechanisms’) and reacting on them as a whole supports this.
- *Distributed Information Storage and Processing* [29]: The pheromone field is stored close to where the information it integrates is generated, and close to where it will be used. Deposits of pheromones only propagate in the vicinity to where they are generated and are only needed by nearby ants.
- *Decentralised Control* [29,4]: Local decisions are made without requiring centralised reasoning or control, and on the basis of nearest-neighbour interactions. Both ant behaviour and pheromone field maintenance are decentralised. Deposits contribute to the field only in their immediate vicinity.
- *Handle Dynamic Situations* [29]: The architecture used to construct and maintain the field is able to incorporate changes rapidly into the field. Under continuous reinforcement, the pheromone field strength stabilises rapidly, as a concave function of time. Thus new information is quickly integrated while obsolete information is quickly forgotten. The coordination is *adaptive* because of mechanisms like exploration, reinforcement of up-to-date information, and evaporation of old and not reinforced pheromones [35,32]. As such, a simple reconfiguration of the ant society is possible (join and leave) without disturbing the system.
- *A single ant is very small compared to the whole colony* in terms of resource usage and impact [4]. The reason for this that an ant, located on a spatial location, only has a limited influence on the system because its sensing and acting is restricted to a very small environment, i.e. its neighbourhood. As such, the approach is *robust to failure of ants*.
- The *global wanted behaviour emerges from the interaction* through pheromones. For example, in [28] the authors emphasise the critical role played by ant communication. In fact, each ant is complex enough to solve a single sub-problem but the global (routing optimisation) problem cannot be solved efficiently by a single ant. It is the interaction between ants that determines the emergence of a global effective behaviour. Ants achieve such behaviour as a group without direct communication or complex reasoning. The latter indicates that the problem solving power resides in the interactions instead of inside the ants’ reasoning.
- Pheromones support *information spreading/distribution* [32]. The pheromone is an *information carrier* for the following information:
 - Spatial information: direction-to, direction-of path

- Most up-to-date info (strongest)
- Other info that is embedded into the pheromones.

The *environment is the distribution mechanism* for information [32]. This shared physical environment participates actively in the system's dynamics [29].

- *Agents control the information distribution*: The environment is responsible for storing, evaporating, and to some extent distributing information. However, which information, where and how often information is dropped is still the responsibility of the agents. The information structure (i.e. pheromones) in the environment is constructed by the agents, not the environment. As such, agents can still achieve their coordination task rather simple by following the pheromones, i.e. a kind of “red carpet”. However, the agents themselves have to supply this red carpet. Other approaches such as Gradient Fields let the environment automatically built the field structure, agents do not control the information distribution.
- Typically pheromones are used for paths which entities of the system can form and follow. Therefore, *spatial properties such as routing* and the optimisation of those routes can be achieved [35]. *Attraction to specific locations and to move in a specific direction* according to a strength property of the observable information in the environment is achieved [32].
- Existence of information is limited over time because of evaporation unless the information is reinforced by the ants [32]. As such *truth maintenance* is supported.
- Pheromones are stigmergic interactions that give rise to a robust self-organising system-level behaviour that *rests on the feedback between the dynamics of the individual agents and the processes that manipulate information in the environment* [33]. There is positive and negative feedback given through the pheromones. Positive feedback occurs between agents. One agent drops pheromone, and another agent observes this and reacts to it, possibly dropping another pheromone to reinforce that the information present is still useful. Negative feedback occurs between the environment and the agents. The evaporation of pheromones is negative feedback to gradually eliminate older and probably obsolete information [28].
- The ants find their way even if introducing *obstacles* changes the landscape [4]. The underlying mechanism of the adaptive optimisation is shown in the following scenario. Figure 3 illustrates this.

Figure 3A displays the initial scenario. The ants move on a straight path that is marked with pheromones. The path connects the food source with the nest. Ants probabilistically prefer to move into a direction where the pheromone trail is strongest. When the existing trail is obstructed (figure 3B) the ants arriving at the obstacle face a random choice of going left or right. Without any pheromone information available the flow of ants splits roughly in half, going around the obstacle. The random choice explores both options (figure 3C). On the shorter path the ants arrive faster back at the original trail permitting a faster flow between the nest and the food source. Statistically, a higher rate of pheromone deposits on the shorter path than on the longer one results. Thus, taking evaporation of pheromones into account, the shorter path reaches higher pheromone strength, attracting more ants until all ants are recruited to the shortest path (figure 3D).

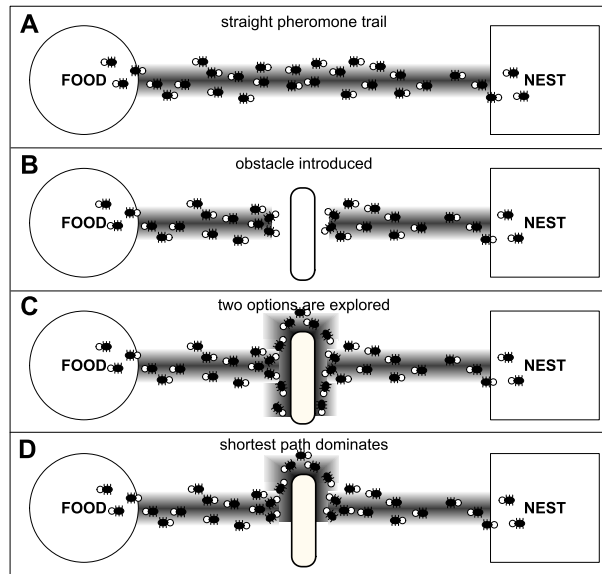


Figure 3. Digital Pheromone Paths can handle obstacle avoidance with shortest path.

4.5 Related Mechanisms/Patterns

Variations

- *Use a More General Interpretation of Pheromones.* Pheromones are not limited to carry only information on their strength. Some examples:
 - Routing tables as pheromones [28]: To route network packets routing tables that indicate the probability of choosing a outgoing network connection can be reinforced by increasing the probability when ants explored it to be good, and decreasing or evaporating the other paths. Each node can also have a statistical model of the traffic distribution and update that as pheromones. While exploring, the ants adaptively build probabilistic routing tables and local models of the network status using indirect and non-coordinated communication of the information they collect.
 - Next to a numerical strength value, each deposit of a pheromone at a particular location can specify a number of other data fields (called tags in [33]). Only the strengths of all deposits at the location that specify the same tags are added together.
 - Products in manufacturing control can be scheduled, planned, and routed through the network of processing machines using loading profile data on each workstation of a factory and updating them as pheromones [32].
- *Use of Multiple Pheromone Types or Flavours* [30,29,36,33,4]. Combinations of multiple pheromones can be used. There are two ways in which pheromones can be considered a different type:

- Different semantics: Different flavours may reflect different features of the environment which have different semantics.
- Different dynamics: Different flavours with the same semantics (e.g. all generated by the same feature) may differ in their evaporation or propagation rate or threshold, thus having different dynamics.

This variation is closely related to letting pheromones contain more than only the strength data because a different type can be established by adding an additional type field to the pheromone [4].

- *Use of Multiple Ant Types.* AntNet [28] is composed of two sets of homogeneous mobile ants, called forward and backward ants. Forward ants explore the network to find the destination. Backward ants return on the explored path to update the pheromones (routing tables and statistical traffic models). Packets then follow the routing tables stochastically just like the forward ants.
- *Directed Pheromones.* Pheromones that indicate a direction, e.g. the direction of the path to follow [30]. Another example is a variation on the propagation of pheromones [4]. Often it is necessary to know at a location from which neighbours a pheromone has been propagated. To achieve this, all pheromones carry a data slot “direction”. The value in the slot is changed during propagation from one place to the next. Only the pheromones propagated from the same place aggregate. The different values in the “direction” slot may be interpreted as specifications of a pheromone type.
- *A Pheromone Pump* [34]. Agents can start and stop what is called a pheromone pump. A pheromone pump resides in a location and continuously deposits a pheromone of a particular flavor. As such pheromones can be automatically deposited from locations where an agent is not present.
- *Using History in Pheromone-based Decision Making* [29]. An agent’s movement through the graph of locations should balance several factors. A strong field gradient enables deterministic hill climbing that the walker should exploit. However, a weak gradient may result from noise in the system. In this case, it does not provide reliable guidance. We would prefer that the agent continues moving in the general direction of its previous steps if there is one, and otherwise that it explores more broadly. Balancing deterministic hill climbing and stochastic exploration. Models of actual ant behaviour usually restrict the ant’s ability to smell pheromones to some angle on either side of its current orientation or probabilistically preferring the pheromones in direction it is currently heading. To take this one step further, an agent maintains an exponentially-weighted moving average of its past headings and modulates the relative strengths of the pheromones in its vicinity by a measure of the angular alignment between each candidate direction and the current value of the heading history.
- *Ghost Agents* [29,4]. Use of Ghost agents to converge to a stable path before the real-world entity moves. Use an agent that represents the real-world entity and moves when that entity moves. And use ghost ants that this agent sends out in front of it. Ghosts move as fast as the network among locations can carry them. They can do “what-if” explorations that physical entities could not afford. Because they move faster than physical entities and their agents, they can look ahead to plan an agent’s next steps. As an agent moves, it continuously sends out ghosts. The

interaction of the ghosts forms the path, which is being constantly revised to accommodate dynamic changes in the environment. Experiments in [29] show this path formation dynamic to be extremely robust and adaptive.

For example, in manufacturing control this can serve in predicting future load profiles of machines. Ghost-ants are regularly created by Workpiece-agents, i.e. the agent moving with the physical product to manufacture. A Ghost-agent emulates a Workpiece-agent in its run through the pheromone infrastructure and represent one possible future of its Workpiece-agent. Encountering a choice between multiple paths, a Ghost-agent samples the same pheromones as the Workpiece-agent and then it takes a similar routing decision. When a Ghost-agent meets a Processing-agent that controls a machine it requests a simulated processing according to its internal state. It immediately receives the simulated new processing state, which it assumes as its new state. Then, the Ghost-agent moves on to the next place and repeat this until it reaches the end of the manufacturing process. While moving, the Ghost-agents provide input to a 'prediction' pheromone which is differentiated by one additional data slot which indicates the current processing state carried by the ghost-agent. There is no propagation. The strength of a 'prediction' pheromone at a location is linked to the probable future local load of workpieces in the state specified by the pheromone and is used when taking routing and scheduling decisions for products. A global parameter determines the rate with which Ghost-agents are created with by Workpiece-agents. This is an important parameter because the more often a workpiece-agent generates a Ghost-agent, the better the quality of prediction.

- *Application Specific Filtering* [4]. Introduce an automatic application-dependent change of pheromones during propagation. The simplest case of application-specific filtering is given when the propagation of a pheromone is blocked on a location depending on the presence of specific agents, or by the presence of specific other pheromones, or another location-specific state. Also, arrival of a propagated pheromone may trigger the generation of other pheromones.
- *Active vs. Passive Pheromone Propagation* [4]. There are two approaches when sharing information through pheromones:
 - *Active Pheromone Propagation*: agents move through the pheromone infrastructure and refresh pheromones according to the local context. An agent may ignore the propagation mechanisms of the pheromone infrastructure and spread its information actively. The agent is in control of what information is put where. Active Pheromone Propagation is more selective and dynamic, because the agent reasoning process may incorporate the current situation and the current local context.
 - *Passive Pheromone Propagation*: the provider agent remains stationary and does not have to care about how the information is spread. The deposited pheromones are propagated to neighbouring locations by the propagation mechanism of the pheromone infrastructure. It is outside of the control of the provider how the information is adapted to the local context. In general, there is no adaptation at all. But the propagation may also use Application Specific Filtering (see above).

colonies to solve discrete optimisation problems. ACO are algorithms optimising problems modeled by a graph. Virtual ants travel across such graph while generating solutions, and update routing tables on their way back depending on their degree of success. More specifically, it has been used for a number of optimisation problems [28]:

- *Traveling salesman problem.*
 - *Quadratic assignment problem.*
 - *Routing, Load balancing, connection management and diagnostics in telecommunication networks.* Information propagation delays, and the difficulty to model the whole network dynamics under arbitrary traffic patterns, make the general routing problem intrinsically distributed. Routing decisions can only be made on the basis of local and approximate information about the current and the future network states. As such, a decentralised routing mechanism is needed.
- *Simulation of ecological processes* such as an ant colony (e.g. AntFarm [30]).
 - *Google Indexing Algorithm* [38]. Google indexes based on some kind of ant pheromone trails. If links represent a dropping of pheromone then the pagerank algorithm is just following the trails laid down to the tastiest morsels, i.e. the more links to a page the higher it indexes.
 - *Manufacturing Control* [32,4]. The products are scheduled, planned, and routed through the network of processing machines using loading profile data on each workstation of a factory and updating them as pheromones. It is a variant that is used here, i.e. ghost-ants are send out by each product currently in progress.
 - *Controlling and Coordinating Swarms of Unmanned Military Vehicles* [34,29,33]. This can be grounded vehicles or unmanned air vehicles (UAVs) in several military operations such as surveillance missions, target acquisition and tracking, and air combat.
 - *Mobile Ad-Hoc Network Management (MANET's)* [39].
 - *Semantic Data Mining.* In [31], multiple stigmergy mechanisms are combined. The pheromone foraging mechanisms is used for identification of relations between concepts found in textual data. Another example is a swarming agent architecture with multiple pheromones for distributed pattern-detection and classification in an active surveillance system [33].
 - *Resource Allocation for Hyperactive agents* [40]. Pheromones are used for regulating local agent activity in support of improved system-level performance. Agents that update their local configuration more frequently than the system's communication latency permits can initiate thrashing. Some degree of agent exploration is desirable when a system is changing, but when it is stable, and as firm deadlines approach, the system should shift to a focus on exploitation. So called 'no-action pheromones' are used to compare to a threshold to decide if an agent should be active or not. Demonstrated on graph coloring, air mission scheduling, and distributed document clustering.
 - Other case studies mentioned [28,32]: *graph colouring, traffic systems, smart power generation, etc.*

5 Pattern 2: Gradient Fields

Also Known As: Computational Fields, Co-Fields, Morphogen gradients [10], Field-based coordination, Force Fields [9].

5.1 Context/Applicability

A solution is needed to coordinate multiple autonomous entities in a decentralised way to achieve a common and globally coherent goal. The coordination mechanism has to be robust and flexible in the face of frequent changes.

More specifically, the autonomous entities are situated in an environment (physical or logical) structure representing the problem to be solved. Some kind of spatial movement of the autonomous entities is required or information about the spatial location of some entities has to be exchanged.

Local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. For example in network routing [28], information propagation delays, and the difficulty to model the whole network dynamics under arbitrary traffic patterns, make the general routing problem intrinsically distributed. Routing decisions can only be made on the basis of local and approximate information about the current and the future network states.

5.2 Problem/Intent

- *Spatial Movement:* How to adaptively orchestrate in a decentralised way the spatial movement of a large set of agents in large-scale distributed systems [9,11,10]? As such global *Pattern Formation* can be achieved, e.g. letting agents meet somewhere [41], distribute agents in an environment according to specific spatial shapes or patterns [42], or simply letting them efficiently move in an environment without interfering with each other and avoid emergence of traffic jams [43]?
- *Structure Formation:* How to adaptively self-configure a modular structure achieving the desired shape/structure (e.g. modular robots) [10]?
- *Routing:* How to achieve routing for communication messages, agents, etc. [10]?
- *Integration of Contextual Information:* How to provide agents with abstract, simple yet effective contextual information from various sources supporting and facilitating the required motion coordination activities [9]? How to make agents context-aware? In particular, how to give agents spatial-awareness [11] (e.g. distance to a source, direction to a source, in which region of the environment is the agent [10])?

5.3 Forces

- *Explore vs. Exploit:* In order to be adaptive the solution has to explore sufficiently compared to only exploiting already known information. Otherwise the approach can get trapped in local optima or never find new targets at all. However, too much exploration may result in an approach that is very inefficient.

- *Centralised vs. Decentralised*: A decentralised solution often requires a huge amount of communication and coordination overhead which a centralised solution has not. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context. A centralised solution also has a huge amount of communication overhead if the information needed to control the system is intrinsically distributed and a lot of it has to be aggregated to the central node (e.g. large-scale heavy loaded (network) systems). Often, a model of the complete state of a system cannot be obtained at all. As such, decentralised control is the only alternative.
- *Optimality vs. Robustness/Flexibility*: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. However, an optimal solution only exists with respect to a static situation, which is never reached in the face of frequent changes. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible. For example, in motion coordination [10] one can give each agent a map of the environment to follow. However, in a dynamic environment this map has to be recomputed each time something changes. As such, calculating an optimal solution for a map becomes obsolete with every change, which occurs frequently.
- *Responsibility of Environment vs. Agents*: Coordination often needs complex processing and communication. There is a trade-off to make the agents themselves or the environment in which they are situated responsible for this processing and communication. For example, in motion coordination [12] one can provide agents with a map of the environment and some kind of communication channel to other agents and coordinate by reasoning, interpreting the map, and deciding what to do and communicating with other agents explicitly. In a dynamic environment, this results in brittle, static and non-adaptive coordination. On the contrary, if the environment itself would represent the needed context information expressively by transparently processing and distributing the needed coordination information towards the agents, agent would trivially use that information as a kind of “red carpet” which, when followed, achieves the coordination task and avoids complex processing within the agents. Such coordination can be more dynamic and adaptive because the source of changes, i.e. environment, also supplies the coordination information.
- *Greedy vs. Focussed*: A “greedy” approach to coordination disregards that a small sacrifice now, i.e. not exploiting a piece of coordination information, could possibly lead to greater advantages in the future. However, it is a general drawback of distributed solutions, where the possibility of globally informed decisions by distributed agents is often ruled out due to the need for efficient and adaptive coordination [12].

5.4 Solution

Inspiration. The Gradient Field coordination mechanism takes its *inspiration from physics and biology*. In physics [11,12,9], the same mechanism can be found in the way masses and particles in our universe adaptively move and globally self-organise their movements accordingly to the locally perceived magnitude of gravitational/electromagnetic/potential fields. The particles follows the “waveform” of the fields (see figure 5). In biological organisms, a coherent, reliable and complex behaviour is achieved from the local cooperation of large numbers of identically “programmed” cells [10]. In

particular, chemicals are diffused among cells and cells are driven in their behaviour by the locally sensed gradients of diffused proteins (“morphogen gradients”). Morphogen gradients is a mechanism used to determine positional information and polarity. For example, in the *Drosophila* embryo, cells at one end of the embryo emit a morphogen (protein) that diffuses along the length of the embryo. The concentration of different morphogens is used by other undifferentiated cells to determine whether they lie in the head, thorax, or abdominal regions to achieve for example wing and limb development.

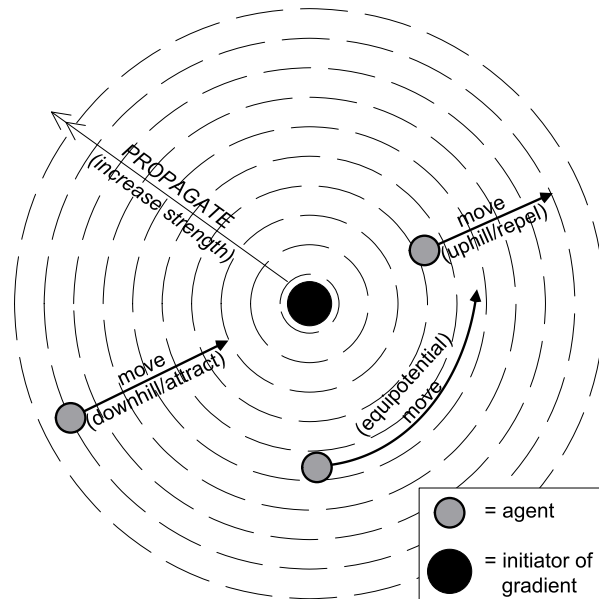


Figure 5. A classic gradient field with propagation direction and agent movements shown.

Conceptual Description. To use this as a decentralised coordination mechanism in software systems such a gravitational/electro-magnetic/chemical field has to be translated into an artificial data structure representing the `Gradient Field`, i.e. a computational field or Co-Field [5,44,45,9,12]. Figure 6 illustrates the conceptual structure of such a solution in UML class diagram notation. A `Gradient Field` is data structure which is spatially distributed, as `Gradient Parts`, over `Locations` in the `Environment`. Each field is characterised by a unique identifier, the necessary contextual information such as a location-dependent numeric value (representing the field strength in that location), and a `Propagation Rule` determining how the numeric value should change in space.

A gradient field is started, initiated, or injected into the environment from a certain “source” location by a `Gradient Initiator` (i.e. a `Location` itself, an `Agent`, or some other entity in the system) conveying some application-specific information

about the local environment and/or about the initiator itself [9]. The Environment makes sure that the Gradient Field is propagated, according to its Propagation Rule, from the starting location to the neighbours of that location (typically increasing the strength of the gradient, initially set to zero; decreasing gradients are also possible). In turn, the neighbouring locations modify the strength and re-broadcast the gradient to their neighbours which is repeated until the gradient has propagated far enough. Each intermediate location stores and forwards only the gradient part with the minimum strength value it has received for that particular gradient field. As such a “waveform” gradient map is formed in the environment which conveys useful context information for the coordination task.

Agents, situated on a location, observe and follow the waveform (deterministically or with some probability) to coordinate their movement with respect to the gradient initiator. For example, in figure 5 agents move downhill (attracted by the initiator), uphill (repelled by the initiator), or on an equipotential line (equal strength around initiator).

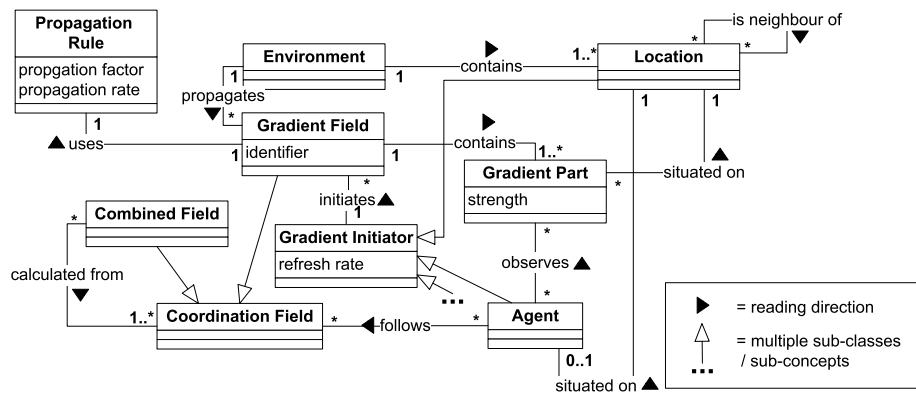


Figure 6. A conceptual model of the gradient coordination mechanism.

The gradient field mechanism can be schematised as follows [5,9,11,12]:

- The *environment is represented and abstracted by “computational fields”*, generated by the agents themselves and/or by some infrastructure in the environment, and spread/propagated across agents and/or the environment according to propagation rules. The key idea behind these fields is to make agent context-aware by providing them, depending on their location, with a locally accessible perspective or “view” on the global situation of the system to facilitate the required motion coordination activities. Note that, this context-awareness is mostly related to spatial-awareness, e.g. distance and direction to the source [11]. Of course, other information can be embedded inside the gradient.
- The coordination policy is realised by letting the agents move locally *following the “waveform” of these fields*, the same as a gravitational particle moves according to the local gravitational field. Agents can autonomously decide whether to follow the suggestions made by the gradient fields or not.

- Environmental *dynamics* and movement of entities *induce changes in the fields' surface*. For example, when the initiator of a gradient field moves, the field -with some propagation delay- has to be updated in the environment to reflect that move, i.e. the 'new' field has to start from the new location of the initiator. As such, a feedback cycle is formed that consequently influences how entities move.
- This *feedback cycle* lets the system (agents, environment and infrastructure) *self-organise*. A globally coordinated and self-organised behaviour in the agent's movements emerges in a fully decentralised way due to the interrelated effects of agents following the fields' shape and of dynamic field re-shaping [9,11]. Complex movements are achieved not because the agents' will, but because dynamic reshaping of the surface.

However, the achievement of an application-specific coordination task is rarely relying on the evaluation, as it is, of an existing computational field. Rather, in most cases, an application-specific task relies on the evaluation of an application-specific Coordination Field [9,11,12]. This coordination field can be an existing gradient field but typically it is a Combined Field, calculated as a combination (e.g. linear) of some of the locally perceived fields or other coordination fields. The coordination field is a new field in itself, and it is built with the goal of encoding in its shape the agent's coordination task. Once a proper coordination field is computed, agents can achieve their coordination task by simply following the shape of their coordination field. As such, more complex motion patterns such as diffusion, birds' flocking, ants' foraging, bee dances, to mention a few examples, can all be easily modeled with fields (i.e. in terms of agents climbing/descending a coordination field obtained as the composition of some computational fields) [11].

Parameter Tuning. Every gradient field has a number of parameters, or "settings":

- *Propagation Factor*: the amount that is added to or removed from the gradient strength at each propagation step.
- *Update Cycle Times*: the regular times at which updates are done:
 - *Propagation Rate*: rate with which a gradient propagates.
 - *Refresh Rate*: a dynamic gradient has to be updated regularly to reflect changes of the initiator. This rate determines how often this occurs. Note that in an event-based solution such an update can be triggered by a change and not at a certain rate.
- *Initial Strength of Gradient*: strength of a gradient at the source location when it is initiated.
- *Propagation rule parameters*: each propagation rule can have its own parameters, other than the propagation factor. For example, using another propagation factor at a distance of X from the initiator.

To facilitate tuning of parameters some formal models of gradient fields can be useful. For example, in [11], the authors model field-based coordinated systems in terms of a dynamical system. However, in situations where agent movements do not occur in an open and continuous space but are constrained by some environmental conditions, a dynamical system description can be more complex or even practically infeasible.

Infrastructure. To support fields' propagation a proper infrastructure or middleware is required. This middleware can be based on an external server in charge of storing fields' values, but it can also be embedded in a distributed environment topology or even in the agents themselves and rely on an epidemic communication schema. The final shape of the field surface will be determined both by the field's propagation rule and by the infrastructure topology.

If the environment in which the agents are situated is responsible for storing and propagating the gradient fields then there are two approaches to construct the needed infrastructure:

- Layer on top of an existing Middleware providing support for [9,11]:
 - Data storing: simple storage mechanisms to store field values
 - Communication: to propagate field values to neighbouring peers
 - Basic event notification and subscription mechanisms: to notify agents about changes in field values, to enable agents to select those fields in which they are interested, as well as promptly update the distributed fields structure to support dynamic changes
 - Mobile-code services: to dynamically configure field-propagation algorithms and coordination fields composition rules
 - Localisation mechanisms: to discover where agents are
- Novel Middleware specifically for Gradient-Fields, e.g. TOTA [11].

This is a trade-off between efficiency and reuse of existing middleware which handles a lot of low-level details for you. Implementing the supporting infrastructure as an additional layer over an existing interaction middleware is not the most efficient solution: a mismatch between gradient mechanisms and the mechanisms used in the middleware may introduce computational and communication inefficiency; and a mismatch between programming abstractions of middleware and those of fields may introduce complicated implementations.

Characteristics.

- Typically, following the gradient field downhill is the *shortest path* towards the initiator of the field [10] because only the gradient parts with the minimum values received for that gradient field are propagated.
- *Static field versus dynamic field [9,11,12]:* Next to the dynamic re-shaping of fields, some fields can also be static. A field is static if once propagated its magnitude does not change over time. These fields are mostly used when the initiator of the field is also static.
- The structure of the environment in which the agents are working should reflect the current “problem” the agents are working on and the gradient structure and distribution should guide the agents to the current “solution” to that problem.
- *Spatial Context Information is distributed:* Gradients support *information spreading/distribution*. The *environment is the distribution mechanism* for information and participates actively in the system's dynamics. Gradient-Fields mainly deliver spatial information such as the direction or distance to a gradient initiator. However, gradients can also embed any other necessary information.

- *Feedback Cycle [5]*: Feedback is given by the fact that gradients can change when changes occur in the environment or when the agent that emits the gradient decides to move or change the gradient. Other agents or gradient-emitting entities can then take that change in the gradient into account and react on it by for example changing its own gradient info. As such a feedback cycle is established to self-organise.
- *Handle Dynamic Situations - Robustness - Openness [9,12,11]*: The contextual information is provided in a robust and adaptive way. The feedback cycle between agents through dynamically changing gradients in the environment makes this possible. As such, changing characteristics of the environment and changes in the number and identities of agents can be handled. This openness of the coordination mechanism makes it intrinsically robust, in that for example, if a component breaks down or gets disconnected from the system, the others can autonomously and dynamically re-organise their interaction patterns to account for such a problem.
- *Scalable(????)*: Due to the intrinsic openness of the coordination mechanism solutions can become scalable in problem size also. For example, in [10] the results indicate that the approach scales linearly with the number of coordinated agents (increase in agents = equal decrease in performance).
- *Decentralised - Emergence [9,11]*: A global self-organised motion pattern emerges in a fully decentralised way due to inter-related effects of agents following the fields' shape and of dynamic fields re-shaping and re-propagating due to agents' movements. The goals that are accomplished are not due to single agents, but due to the system as a whole without any central controller.
- *Simple Agents - Complex Environment*: Field-based approaches delegate to a middleware infrastructure in the environment the task of constructing and automatically updating the gradient field [11]. As such, the environment makes sure that not too much computational and communication burden is imposed on the agents themselves by automatically providing a dynamically adapting and propagating coordination structure that is immediately usable by agents [12]. The context is represented expressively as gradient fields, i.e. a kind of "red carpet", which represents how to achieve a coordination task by simply following the field. The coordination is achieved with very little effort by agents. On the contrary, when agents do not have such a support and use a global static map of the environment then complex algorithms and communication protocols are needed to elaborate, interpret and decide what to do next.
- *Greedy Approach [11,12]*: A weakness of field-based approaches is that they are "greedy" because of the strictly local perspective in which agents act. Agents disregard whether as small sacrifice now, climbing a field hill instead of descending it - could possibly lead to greater advantages in the future.
- *Explore versus Exploit*: Related to the greediness of gradient fields is the trade-off between exploring and exploiting a gradient. Agents can observe and follow the waveform deterministically or with some probability. That probability can be tuned to reach the most suitable degree of exploration, i.e. increasing the probability decreases the exploration. As such, the greediness can be tempered.
- *Diversity of Information*: Various types of information from various sources are possible in multiple gradient fields.

5.5 Related Mechanisms/Patterns

Variations

- *Propagation Inhibition or Selective Propagation [11,10]*: The spatial-awareness promoted by gradient fields is mainly related to the direction and distance to the source of the field. However, fields allow also dealing with spatial concepts in a much more flexible way by exploiting a much more physically-grounded concept of space. For instance, one can bound the propagation of a field to a portion of space by having the propagation procedure to check conditions such as the local spatial coordinates, the gradient type or its strength to decide on further propagation or not. Thus, locations in the environment can act as barriers/inhibitors to specific gradients, or as obstacles around which the gradient must travel.
As such, one can achieve so called *region selection*. If a leader propagates a gradient which is inhibited in its propagation when it reaches a maximum strength, one can create (approximately circular) regions of controlled size and have agents recognize the specific circular region they are in.
- *Multiple Types of Fields [11,9,12]*: As mentioned earlier, multiple gradient instances can be combined in a coordination field to follow as a whole. All used gradients are typically of the same type. A logical extension is using different types of fields depending on the specific motion pattern to enforce. As such, they can be propagated and combined in coordination fields according to field-specific rules.
- *Adaptive/Evaporating Gradient Fields [10]*: We can have the gradient value stored by any location lose significance if not constantly reinforced. The result is that old gradients disappear gradually when the initiator moves, disappears, or no longer injects the field. This is in contrast to the sudden removal of a field when stopped. This is closely related to the evaporation of pheromones.
- *Chain of Gradient Fields [10]*: Agents start diffusing specific types of gradients, and other agents sensing that gradient start diffusing a new type of gradient as a reaction. Repeating this process with new types of gradients being propagated in different phases, allows to incrementally coordinate with a chain of gradient fields.

Other Coordination Mechanisms. *Digital Pheromone Paths* is a specific instance of *Gradient Fields* where a number of small fields or pheromones aggregate into a gradient path and evaporate over time [11,34]. Also pheromone paths are constructed explicitly and, as such, agents also have to discover targets explicitly. Gradients make targets immediately and automatically (by the environment) visible through the presence of a gradient field for that target. In addition, pheromones constitute a memory of the recent past, while gradient fields are instantaneous.

5.6 Examples/Known Uses

- *Spatial Shape Formation [10]*: The problem of having a multitude of simple mobile computational “particles” (i.e. sorts of small mobile robots) self-organise their global shape so as to obtain a variety of spatial configurations or shapes. Many algorithms are used:

- Barycenter: election algorithm to elect the center of gravity, i.e. given n particles in space, the barycenter is that particle that minimizes the sum of the distances to all n particles.
- Circle: first phase elects barycenter, then barycenter emits circle gradient in second phase and particles move to where circle gradient has distance R from barycenter (determined based on gradient strengths).
- Ring: building upon the circle algorithm the particles on the circle positions emit a second gradient which attracts particles to a distance T from it. As such, at distance T from the circle, the border of a ring is formed.
- Lobes: Circle algorithm extended to move particles away from crowded parts on the circle.
- Polygons: Elect n (= number of corners in polygon) equidistant particles on circle positions and let them execute the lobe extension algorithm.

Other spatial phenomena, such as birds' flocking in formation and bee dances, can all be easily modeled with fields and all have practical application in mobile computing scenarios [11].

- *Urban Traffic Management [9]*: Enable the coordination of the respective movements of the vehicles in a city. It is assumed that the city is provided with an adequate computer network embedded in its streets/corners, and cars which can communicate with their nearest neighbours via a wireless network. Each host represents a meaningful zone of the city such as a traffic light or roundabout and the network topology mimics the street topology. Global coordination needed is:
 - move in the city by avoiding traffic jams or queues
 - overall load balancing of the city traffic
 - need for a group of cars to meet with each other at the most suitable location
 - evacuate in the most efficient way specific portions of the city
 - move in the city accordingly to specific formations, e.g. consider police cars in need to properly monitor the city

In [9], 3 simple fields are used: Street-Corner Field (SCF) generated by every street/corner increasing in magnitude with distance to attract cars that need a route through that street/corner; Presence Field (PRES) generated by cars to signal its presence; and Traffic field (TRF) which measures the amount of traffic in a street/corner by summing the total number of presence fields with smallest value, determining the cars present, and normalising that number to the dimensions of the street/corner (see figure 7). This traffic field is adjusted dynamically, depending on car movements. As such a coordination field to achieve the coordination task of avoiding traffic or queues is calculated as follows: $CF = \min(SCF_1, SCF_2, \dots, SCF_n) + \lambda * TRF$. The first term attracts cars to the streets/corners in their route, and the second term avoids highly congested streets/corners. λ is a parameter indicating how sensitive the coordination is to traffic congestion (high=very sensitive).

- *Reconfiguring Modular Robots' shape [9,13]*: Modular robots are made up of autonomous components flexibly connected with each other, and globally enabling a reconfigurable shape. Computational fields are used to achieve the desired shapes.
- *Control of Autonomous Characters in Video Game "The Sims" [9] and Quake 3 [12]*: In "The Sims" The behaviour of the non-player characters is directed by a

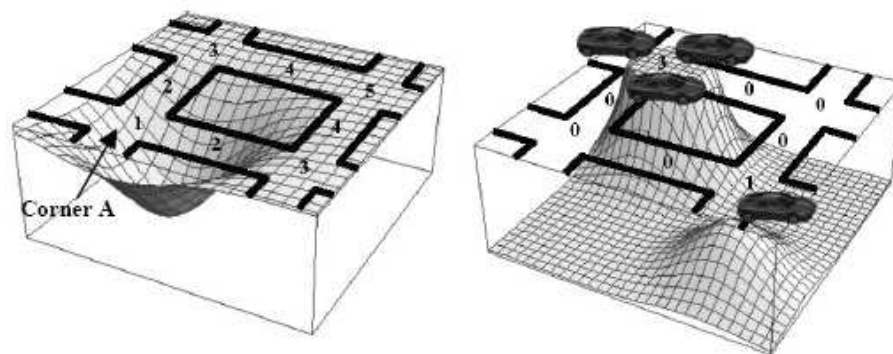


Figure 7. (left) Street/corner Fields: ‘corner A’ generated field; (right) traffic field

“happiness landscape”. In the Quake 3 case, the goals of bots’ coordinated movements can be various: letting them to meet somewhere, distribute themselves accordingly to specific spatial patterns, surround an enemy, or simply move in the environment without interfering with each other.

- *MMASS formal model for multi-agent coordination* [9]: This model represents the environment as a multi-layered graph in which agents can spread abstract fields representing, different kinds of stimuli.
- *Tourist Movement in Museum* [12,11]: Exemplary pervasive computing problem of tourists visiting a museum assisted by agents running on hand-held computers to avoid crowds or queues, find guides, etc.
- Other possible application domains [12]: The above scenarios and the associated motion coordination problem are of a very general nature in a lot of scenarios ranging from other pervasive computing applications (such as traffic management and *forklifts activity in a warehouse*), to internet-scale scenarios (such as software agents *exploring the web*, where mobile software agents coordinate distributed searches by moving on various websites).

6 Pattern 3: Market-based Control

Also Known As: Market Control, Market-Oriented Programming [14].

6.1 Context/Applicability

You need to coordinate multiple autonomous entities in a decentralised way to achieve a common and globally coherent goal while sharing a set of scarce resources as efficiently and fair as possible. The coordination mechanism has to be robust and flexible in the face of frequent changes.

Local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. For example in network routing [28], information propagation delays, and the difficulty to model the whole network bandwidth usage under arbitrary traffic patterns, make the general resource (bandwidth) allocation problem intrinsically distributed. Some locally available information is needed indicating the global usage of resources.

6.2 Problem/Intent

- *Resource Allocation:* How to do efficient resource allocation and control [6,15,16,14] of computer systems in a distributed and decentralised manner?
- *Integration of Resource Usage/Need Information:* How to have locally available information about the global usage of and need for resources?
- Depending on what the ‘resource’ is a number of problems can be solved such as:
 - Resource = spatial or temporal regions; allows for spatial and/or temporal distribution load balancing [35].
 - Resource = task; allows for task allocation
 - Resource = bandwidth; allows for load balancing and routing in networks
 - Resource = manufacturing device; scheduling in manufacturing control

6.3 Forces

- Centralised vs. Decentralised: A decentralised solution such as this pattern often requires a huge amount of communication and coordination overhead which a centralised solution has not. A centralised solution can often optimally control the system. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context. A centralised solution also has a huge amount of communication overhead if the information needed to control the system is intrinsically distributed and a lot of it has to be aggregated to the central node (e.g. large-scale heavy loaded (network) systems). Often, a model of the complete state of a system cannot be obtained at all. As such, decentralised control is the only alternative.
- Optimality vs. Robustness/Flexibility: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. Especially because such a solution relies on local incomplete view about wider system and a local decision can have non-local effects. However, an optimal solution only exists with

respect to a static situation, which is never reached in the face of frequent changes. In addition, optimality requires complete information about state of entire system which is impossible. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible.

- *Responsibility of Environment vs. Agents*: Coordination often needs complex processing and communication. There is a trade-off to make the agents themselves or the environment in which they are situated responsible for this processing and communication. Making the agents responsible allows agents to explicitly reason about and control how information is distributed but sometimes requires complex algorithms. On the other hand, making the environment responsible allows agents to simply be guided by the results in the environment but the agents are no longer in control of the information distribution.

6.4 Solution

Inspiration. The inspiration and metaphor came from natural human economies and market mechanisms, i.e. the free-market economies. In such an economy, goods or resources are allocated to the participants in a decentralised, robust, and self-organising manner. Participants act as buyers and/or sellers of goods by offering to buy or sell a good to other participants for a certain price. As long as the participants act completely self-interested and do not take into account the gain of the others then the market achieves a globally optimal allocation of goods, i.e. Adam Smith’s “invisible hand” that guides the agents in lowering/increasing their prices. Eventually economic theory states that the transaction prices converge to a global equilibrium price, i.e. the market price. Note, that the degree of convergence depends on the market characteristics (e.g. elasticity, shape of demand and supply curves, etc.).

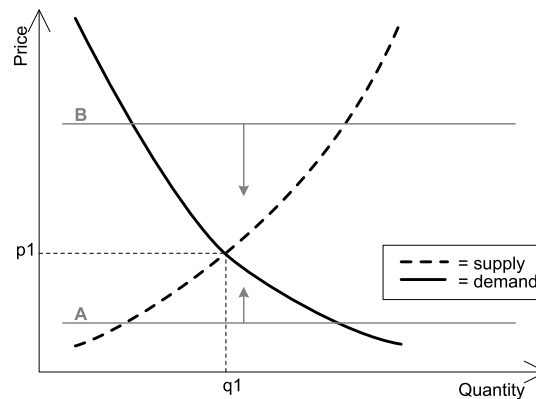


Figure 8. The theory of Supply and Demand.

This price mechanism depends on the evolution of the demand and available supply of goods. Figure 8 illustrates this by plotting the evolution of supply and demand. The demand curve shows the quantity of good that consumers are willing and have the capacity to buy at the given price. The supply curve shows the quantity that suppliers are willing to sell at a given price. If the quantity of a good demanded by consumers in a market is greater than the quantity supplied, competition between consumers causes the price of the good to rise. As such, according to situation A in figure 8 this can both reduce the quantity demanded (because some consumers can no longer afford it) and increase the quantity supplied (because some suppliers may be more interested in selling at higher prices). Similar effects occur when the quantity supplied is greater than the quantity demanded: competition among suppliers can lead to the price falling, which may reduce the quantity supplied and increase the quantity demanded (situation B in figure 8). According to classical economic theories the market equilibrates (see arrows in figure 8): transaction prices approach an equilibrium value (p_1, q_1) where the quantity demanded matches the quantity supplied. As such, an efficient means of societal resource/goods allocation exists. The complex laws of supply and demand are the fundamental building blocks.

Conceptual Description. The aim of market-based coordination in computer science is fundamentally different from the aim of economic theory [14]. In market-based coordination, microeconomic theory is taken as given and serves as the theory for implementation of computational agents. Whether or not the microeconomic theory actually reflects human behaviour is not the critical issue. The important question is instead how microeconomic theory can be utilised for the implementation of successful resource allocation mechanisms in computer systems. Mathematical economics serves as a blue print for engineering similar mechanisms in computer systems. Although decision-making is only local, economic theory, which has an immense body of formal study [46], provides means to generate and predict macroscopic properties such as the equilibrium price and others that can be deduced from that price information [35]. The aim is that computer systems exhibit the same decentralisation, robustness, and capacity for self-organisation as do real economies [15]. In designing a market of computational agents, a key issue is to identify the consumers and producers of the goods to be traded [17]. Various preferences and constraints are introduced through the definition of the agents' trading behaviour. This ability to explicitly program trading behaviour is an important difference from the situation with human markets. Finally, the market mechanism for matching buyers and sellers must be specified (globally for an auction, locally for a direct market).

Figure 9 illustrates market-based coordination conceptually in UML class diagram notation. The system is constructed as a group of `agents` that are negotiating and trading with each other on an virtual market of scarce resources. The `agents` communicate with messages that encapsulate `offers`, `bids`, `commitments`, and `payments` for resources [16]. The `agents`' goal is to acquire the `resources` of a certain `Resource Type` that they need to achieve their individual control action goals [47]. All `agents` start with an initial amount of `Currency` [17]. `Resources` are at each moment in time owned/used by one `agent`. Some `agents` act as 'consumers' or 'buyers' and are

ers are willing to supply more for each price. This results in an equilibrium increase in the supply and decrease in price (similarly the demand curve shifts when agents suddenly need more resources). A globally limited amount of available resources makes the market mechanism a resource allocation mechanism that is supposed to generate an equilibrium distribution of resources that maximises social/global welfare [16].

For example [35], consider a network in which autonomous entities have to find resources to complete an assigned task by moving to suitable hosts. These resources could include access to the CPU, network and disk interfaces, and data storage. A required self-* property is that the entities are allocated efficiently. There is the danger that entities will exploit network resources and cause unwanted resource contention. Agents need the ability to coordinate at a high level to distribute their load evenly across the network and over time. Market-based control can address each of these points. An autonomous entity that arrives at a host will use electronic currency to purchase the resources necessary to complete its task. To coordinate this, information is needed about the resource usage at hosts. A market provides usage information through prices. High prices connote congestion. This gives agents incentive to distribute themselves evenly in the network by choosing another host or wait until lower prices, i.e. less congestion. Another example concerns Task Allocation [48]. The market-based solution to the problem of allocating a task (or tasks) among a large number of agents is solved when agents bids for (part of the) task which are offered elsewhere in the system, i.e. by suppliers.

There are two approaches to establish a market in a computer system, i.e. two possible market structures [49,16,15,14,17,50]:

- *Auctioneer-mediated Markets (Centralised)*: Many applications in literature use this kind of market [15]. The market is mediated by a market institution or an auctioneer agent. Agents send demand/supply functions or bids telling how much they like to consume/produce at different prices. The auctioneer then tries to establish an equilibrium market price vector such that supply meets demand for all resources. This equilibrium is established immediately, i.e. no iterative convergence. Then agents exchange the resources as stated by their bid and the calculated equilibrium price, i.e. buyers only buy at a market price under their bid price, and suppliers only sell at a market price above their bid price. The auctioneer is a central controlling entity.

Multiple kinds of auctions can be used [16]:

- English-auction markets: sellers remain silent while buyers quote increasing bid-prices: bid-prices ascend until only one buyer remains, at which point a transaction occurs.
- Sealed bid double auctions: the bargaining is completed in one shot that would take indeterminate time using iterated market institutions such as continuous double auctions. Sealed bids have the inefficiency of trading away from the equilibrium which can be balanced against the fact that we allow agent to adapt the changes over time.
- Continuous double auctions: bids can be made in multiple iterations or phases to allow buyers adjust their price according to what was done in the previous iteration (bid higher than previous highest bid). As such, the number of iterations is indeterminate and depends on when the buyers stop changing bids.

- First Price protocol: price at which buyers and sellers trade is that of the highest bid submitted
 - Vickrey, or Second Price protocol: price at which buyers and sellers trade is that of the second highest bid submitted.
- *Direct Markets (Decentralised)*: Real bids are used instead of calculating based on demand and supply functions. The market is executed for real and the agents bid and adapt their bidding to the outcomes of the auctions over time. A seller can quote an offer-price at any time, and a buyer can quote a bid-price at any time. A transaction occurs whenever one trader accepts an offer or a bid quoted by another trader. At any time, any trader may quote a new price that supersedes that trader's previously-quoted price.

In this system, the environment consists entirely of the graph of dependencies among different products induced by patterns of joint use, since it is through these dependencies that individual transactions have an effect on one another. Transactions depend on pairwise encounters in which agents exchange their bid or price for a resource (e.g. traders distributed in space only transact with nearby traders). The agents bid and adapt their bidding to their success over time which iteratively balances supply and demand at the equilibrium [16].

Note, that the speed or stability of equilibration in a market can be affected by the type of market. An auction-mediated market typically equilibrates immediately in each auction cycle while a decentralised market needs multiple transactions cycles before the market equilibrates. However, the system should not rely on the operation of any single critical component or sub-system, i.e. a centralised auctioneer [15]. In much the same way that the national market for bread does not collapse when one baker goes bankrupt, so the failure of any one trading agent in a market-based system should result in only a minor impairment (if any) to the overall behaviour of the system, rather than a total breakdown. A truly decentralised market-based solution requires agents with the capabilities to directly bargain, negotiate, and trade with other agents. Such a truly decentralised and robust system is preferred to a failure-prone and inflexible central one. However, firm guarantees that the (optimal) equilibrium price is reached are not always available because less theory is available for decentralised markets compared to centralised markets.

Another important aspect of markets, that is often ignored in decentralised systems from literature, is the need for a real price mechanism. Often, ideas from economics are used as a weak metaphor and there is nothing that approximates to a currency or price mechanism [15]. There should be a meaningful price mechanism: a currency should be available for expressing relative utility, usage, indifference, substitution between resources, and so on. For example, when using contract nets just for negotiating who will use which resource but without the mechanism of price setting, buying and selling, then this is not a real decentralised market-mechanism.

Parameter Tuning. Depending on the specific use of the market mechanism, a number of parameters have to be tuned:

- *Limit Prices*: To maximise the profit and to minimise the spending this limit price may not be too high or too low. The limit price determines the demand and supply curves and as such how and how fast the convergence of the market will evolve.
- *Initial amount of currency*: at the system start each agent needs an initial amount of currency to use in buying resources. This amount has an influence on the limit price buyers can set because that limit price has to be covered by the needed currency.
- *Number of agents*: In systems where the number of agents is not restricted, this number can influence the speed of convergence to equilibrium because market mechanisms work best when the group of participants is rather large. However, not too large for efficiency reasons.
- *Size and Speed of price modifications*: Figure 8 shows that the slope of the demand and supply curves determine how fast a market price changes. For example, a very steep demand and supply curve implies that a small change in the quantity supplied or demanded results in a large change in price towards equilibrium (i.e. a very elastic market in terms of economic theory). From the agent point of view this depends on how fast and how large the suppliers and consumers modify their prices in response to for example repeated failure in acquiring a resource. So depending on what is required tuning these decision parameters is important.
- *Who trades with Who?*: For the decentralised market, the specification of who trades with who is also an important parameter. For example, a distance-limit between agents in a 2D space can determine if they can trade or not. This determines the frequency of pairwise encounters and thus of transactions.

Infrastructure. Because most of the coordination happens directly between agents no real infrastructure support is needed, except for communication infrastructure.

Characteristics.

- *Information Distribution*. Markets deliver to agents information on the resource-usage through the price mechanism. A high price reflects a high demand and/or low remaining supply, i.e. high usage. A low price reflects a low demand and/or high remaining supply, i.e. low usage.
- *Feedback Cycle*. The price mechanism serves as feedback between the economic partners. A high price often implies that agents will start buying less which diminishes the demand and as such the price will reflect this by decreasing. In turn, this lower price changes the behaviour of agents to start buying again and increasing the demand. As such feedback about the global market is given through the price evolution.
- *Agents have all responsibility*. The responsibility of coordination is completely situated with the agents themselves, i.e. no environment-mediated coordination. There is no infrastructure in the environment that takes some responsibilities away from the agents. However, for auction-based markets the auctioneer can be considered as the environment.
- *Suited for Multi-Enterprise Coordination [16]*: In a multi-enterprise situation the mechanism has to make sure that no sub-system owner is forced to expose informa-

tion on matters affecting their own commercial interest. The market-based mechanism only needs exchange of prices between enterprises. That price only reflects the current usage/demand/availability of the resource but no internal enterprise details.

- *Decentralisation - Robustness - Self-Organisation [14,16,15,48]*: Market-based control without an auctioneer allows for truly distributed systems with the same decentralisation, robustness to participant failures, and self-organising properties as real free-market economies. A central auctioneer would be a single point of failure.

A primary advantage of decentralisation is that it offers *robustness in the form of graceful degradation*. The absence of centralisation can lead to systems that show a progressive loss of performance, rather than sudden catastrophic collapse, when individual subsystems malfunction or fail. Often this robustness is a consequence of *self-organisation*: there is no need for explicit reprogramming when a subsystem fails; rather, the remaining subsystems alter their activity to accommodate the change in their operating environment. Sometimes this self-organisation also extends to the initial programming or calibration of the system.

- *Scalability [48,16,15]*: At any time the number of agents participating in the market can be increased. For example, in [16], a market-based approach to network routing implies that no agent needs to know of the existence of more agents than there are links in the paths of the network which makes it more scalable. Often, a larger number of agents and interactions speeds up the market convergence to equilibrium and also classical theory dictates a practically infinite or large number of trading agents to achieve price-equilibration. However, according to [15], market-based systems converging fast to equilibrium are also possible with few agents.
- *Price Information indicates Global and Local Performance*: For example, in [16] a market-based approach to network routing is used. The cost of each network call can be computed from the price information available to the initiator which allows more efficient call charging. Trading resources for some sort of money enables evaluation of local performance and valuation of resources [14] based on the price, so that it becomes apparent which resources are the most valuable and which agents are using the most of these.
- *Stabilisation at Equilibrium - Pareto Optimality [15,18,14]*: For markets to be of genuine use in applications, they should exhibit smooth and fast convergence to the equilibrium. Transaction prices stabilise rapidly at an equilibrium that is predictable from economic theory and which is stable and robust with respect to sudden changes in the market.

Actions of groups of individuals, engaging in simple trading interactions driven by self-interest, i.e. competitively, can result in optimal resource allocation. The control resulting from market-based systems is termed Pareto optimal. Pareto optimality means that no agent can do better in the market place without diminishing the performance of another. This does not necessarily mean that a globally optimal solution is reached. If all utility functions (i.e. agent satisfaction w.r.t. resources) are quasi-linear also, then a global optimum is reached according to economic theory. In the presence of uncertainty the outcome maximises the expected global utility.

- *Not always an Optimal Equilibrium? [15]*: There are indications in theoretical economic studies that the dynamics of some decentralised markets populated by simple

- traders acting purely to serve their own self-interest may converge to stable but highly sub-optimal equilibria, or may exhibit complex (hyper-)chaotic dynamics.
- *Simple Agents/Interactions/Information - Emergence - Efficient [14,15,17,48,18]:* Although it may seem intuitively obvious that some form of ‘intelligence’ is necessary in trading agents, there are results that indicate that very simple agents can exhibit human-like behaviour in markets. The interactions between agents, and the specifications of the agents themselves, can be simple in comparison to the behavioural complexity of the overall system. Thus, the complex behaviour emerges from simple interactions between simple agents. Compared to a centralised approach, relatively little knowledge about the system to be controlled is needed: a single price summarises the demand for each resource. This makes markets an efficient coordination mechanism because agents are rather simple to program and the communication and computation load is minimised.
 - *Adaptivity - Flexibility [14,17,48]:* Market-based control allows to robustly coordinate a distributed real-time response with many elements in the face of failures, delays, an unpredictable environment, varying number of agents (openness), a limited ability to accurately model the system’s behaviour, and a limited available amount of resources to use.

6.5 Related Mechanisms/Patterns

Variations

- *More sophisticated markets [15]:*
 - Spatial structure or segmentation in the market: the traders are distributed over some space, and each trader can only transact with other traders in its local spatial neighbourhood. As such the pairwise encounters on which a direct market relies depend completely on the current locations of the agents [49]. This may impose a slower convergence to the market equilibrium or a segmentation in locally equilibrated markets because some agent groups never encounter agents of another group. Traders can also segment the market according to other conditions than the distance between them (e.g. type of product).
 - Agents engaging in arbitrage: they sell units when the price is high, with the intention of buying them back when price falls.
- *Poor and Rich Agents [14]:* Different amounts of money can be assigned to different agents in accordance with the importance of their projects or tasks.
- *Multiple Markets [16]:* A complex multi-level economy is modelled, i.e. multiple related markets are used. An example on how such markets can be related is when resources on one market are a composite of resources on another market. For example, in [16] there is a market of ‘paths in a network’ and a market of ‘links between nodes’. The paths are composed out of links.
- *Profit Margins:* Agents use so called profit margins in their decisions. There will be no transaction if the profit is not larger than or equal to that margin. Agents can possibly adapt this margin to maximise their profit [15]. If the margin is too low, the supplier may miss out on potential profit when making a transaction. When it is too high, the supplier may miss the opportunity to make transactions with other agents because the wanted price is too high for them.

- *Redistribution of Profit*: All profit is equally redistributed to the consumer agents [17]. This funding policy implies that the total amount of currency in the system stays constant and no money has to be generated to keep the system running.

Other Coordination Mechanisms. Another coordination mechanism that can solve resource allocation is the *Tokens* mechanism. A token then represents the capability and authority to use a certain resource exclusively and the number of tokens that are flowing between agents in the system is limited to the total available amount of resources.

6.6 Examples/Known Uses

- *Manufacturing Control [19,49]*: AARIA is based on a model of the factory as a marketplace. The participants in the market are the workstations that offer to sell certain activities with the product, and the agents representing a job that moves through the factory, has a certain demand of activities and the order in which they should occur. As such a market mechanism can do its job of allocating the workstations to jobs.
- *Power Distribution [14]*: Using market mechanisms for power distribution in residential and industrial energy management settings. Power is the scarce resource to allocate.
- *Routing in Networks [16]*: Market-based approach to call routing in telecommunications networks where agents represent the various network resources (links, paths, and calls) to coordinate their resource allocation decisions. The bandwidth is the base resource. This provides a distributed, robust and efficient means of traffic management. An auction is used to implement the markets: a link market auctioneer selling slices of bandwidth on links between nodes; and path market auctioneer selling paths to call agents which represent a call that has to be routed.
- *Stabilisation of Unstable Structures - Smart Matter [17,18]*: Embedding microscopic sensors, computers and actuators into materials allows physical systems to actively monitor and respond to their environments, i.e. creating smart matter. When a high density of such devices is used, a central controller cannot cope with the amount of information and is a single point of failure. An appropriate decentralised control program is a market-based multi-agent solution. The problem to solve is to maintain a physical system near an unstable configuration. For example, incorporating technology in the design of civil structures that enhances the lateral integrity of the structure during natural excitation such as earthquakes is not new [18]. The control problem is how hard to apply forces on various mass points to maintain the whole system at the unstable fixed structure. The forces are not applied directly but rather indirectly to the structure through the use of embedded actuators that can change the stiffness and/or damping properties of the overall structural systems. In such cases, there is the need to robustly coordinate a physically distributed real-time response with many elements in the face of failures, delays, an unpredictable environment, and a limited ability to accurately model the system's global behaviour.

In the market control of smart matter, embedded actuators are treated as consumers. The external power sources are the producers. All consumers start with a specified

amount of money. All profit that producers get is equally redistributed to the consumers. This funding policy implies that the total amount of money in the system will stay constant. We assume a market mechanism that rapidly finds the equilibrium point where overall supply and demand are equal. This equilibrium determines the price and the amount of power traded. Each actuator gets the amount of power that it offers to buy for the equilibrium price and uses this power to push the unstable structure. The force produced by such an actuator agent is proportional to the power it buys. The demand depends on sensor information the actuators gather. Each produces tries to maximise its own profit given by the difference between its revenue from selling and its production cost. The amount of power available to the system can also be limited. This is then taken into account in the demand and supply.

This achieves stability robustly by focussing control forces in those parts of the system where they are most needed in face of a limited amount of available power.

- Force allocation in an Airjet Paper System [48]: A market-based scheme is suited for allocating and coordinating the actions of many mechatronic systems using analog electronics. In particular, an airjet paper system applying forces and torques for controlled object motion can use a power market to allocate the power to the airjets that need it to achieve the wanted motion as a whole. The total power on the force market and torque market are constrained respectively to the total wanted force and torque.

In general, a market-based solution to the problem of allocating a task (or tasks) among a large number of mechatronic agents (actuators, controllers, sensors, etc.) is solved when each agent bids for part of the total task as determined elsewhere in the system (e.g. by higher level controllers or external inputs). The behaviour of each agent depends on its local information as well as market prices that communicate global information. These agents interact through a small number of markets in order to coordinate their global behaviour.

- *Climate (air-conditioning) Control in Buildings [14]*: Market-based resource allocation for climate control in large buildings in which the cooling power or conditioned air is considered as a resource and agents representing temperature controllers in rooms buy and sell that resource through a central auctioneer that calculates the equilibrium price based on the received supplies and demands. The task is to allocate a resource (cold air) in an office building, given the setpoint temperature of the respective offices. Each office can make use only of a fraction of the available cooling power.
- Other possible cases on distributed resource allocation [14,15]:
 - *File Allocation problem*
 - *Multi-commodity Flow problems*
 - *(Job-shop) Scheduling*; time slots as the resource
 - *Memory Allocation* in an operating system; memory as the resource
 - *Distributing and enforcing pollution limits*; pollution permits as resources with which firms trade

7 Pattern 4: Tags

Also Known As: Labels [51], Social cues.

7.1 Context/Applicability

You need to coordinate multiple autonomous entities in a decentralised way to achieve a common and globally coherent goal in which all agents are stimulated to cooperate instead of act selfishly. The coordination mechanism has to be robust and flexible in the face of frequent changes.

Local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. Some locally available information is needed indicating what is the best behaviour for the collective.

7.2 Problem/Intent

- *Trust mechanism:* You want to develop a mechanism that allows agents to find out from each other how reliable another agent is. Interaction can then be limited between agents that are highly reliable or trusted (i.e. *self-protection*).
- *Formation of Cooperative Groups, discourage selfish behaviour [7,52,53,54]:* You want to develop a mechanism that discourages and/or controls selfish behaviour (e.g. in a P2P file sharing system peers download files without sharing them), and encourages cooperative or altruistic behaviour (e.g. sharing high quality files). All individuals benefit if all act altruistically, but each has an incentive to act selfishly. There is a high level of individual autonomy, i.e. they can behave as they like, and tend to use their abilities to selfishly increase their own utility. Also, new agents can enter the system. So how might strangers who interact only once come to cooperate? How then to devise protocols that will lead to desired cooperative system-level functions?
- *Specialisation within groups [53,54]:* How to obtain specialisation in skills so that they form groups that contain a diversity of skills within them and donating tasks to other more skilled agents occurs often even when the cost of giving is high to the donating agent (i.e. cooperation is high compared to selfish behaviour). As such, agents spontaneously change their specialism (behaviour) if they come to recognise that they might be able to do better following a different role. Then they form internally specialised groups or tribes.

7.3 Forces

- *Centralised vs. Decentralised:* A decentralised solution such as this pattern often requires a huge amount of communication and coordination overhead which a centralised solution has not. A centralised solution can often optimally control the system. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context. A centralised solution also has a huge amount of communication overhead if the information needed to control the system is intrinsically

- distributed and a lot of it has to be aggregated to the central node (e.g. large-scale heavy loaded (network) systems). Often, a model of the complete state of a system cannot be obtained at all. As such, decentralised control is the only alternative.
- *Optimality vs. Robustness/Flexibility*: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. Especially because such a solution relies on local incomplete view about wider system and a local decision can have non-local effects. However, an optimal solution only exists with respect to a static situation, which is never reached in the face of frequent changes. In addition, optimality requires complete information about state of entire system which is impossible. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible.

7.4 Solution

Inspiration. Ideas from the social sciences are less well explored as a possible source of so-called self-* engineering techniques [54]. The main reason is that (human) social phenomena appear to be scalable, self-repairing and self-regulating and often robust - all desirable properties for self-organising information systems. They spontaneously form, and emerge apparently functional structures, institutions and organisations. Much social scientific research has been produced concerning why and how social phenomena occur and social science itself has numerous sub-disciplines, sub-schools, methodologies, and approaches. Also a new area of computational social science has begun to emerge which investigate sociologically motivated computational models. The emergent results and outcomes are observed to gain general insights into mechanisms of social emergence and then relate these to real human societies. However an engineer looking for new techniques to construct self-* systems can use these mechanisms independent of the fact that they resemble real human social phenomena.

One specific social phenomena is the “tragedy of the commons” where, although everybody may greatly benefit from a common resource, it is destroyed through the selfish actions of each person [52]. These sorts of situations are well studied in the social sciences since they occur in many situations in naturally occurring social systems (e.g. over-grazing on a common plot of land or polluting the environment). There have been many computational models to explore ways out of these dilemmas. We explore how one particular solution may be applied. This involves an algorithm that employs the recognition of arbitrary externally detectable attributes, which are called “tags”. We migrate the tag-mechanism from a social science oriented model to a practical application while preserving the desirable scalability and robustness properties.

Another basic idea of tags is that of a kind of ‘cultural group selection’ and the well known social psychological phenomena that people tend to favour those believed to be similar to themselves even when this is based on seemingly arbitrary criteria [54]. For example, people may be able to tell whether another is of the same social group as them by observing the style of clothes of the other (e.g. wearing the same coloured tie) [52]. Such subtle external indicators, where they have no other function, are their tags. Periodically, people tend to copy other people that are performing better or randomly but less probable than copying change their own strategy. As such they change to or select another group to interact with.

Conceptual Description. Tags are observable or externally detectable attributes, labels, markings or social cues that are attached to individuals or agents [51,52]. So, an Agent has one or more Tags. These can be adapted and created by other Agents (Tag Adaptation) and also observed by Agents (Tag Observation). Tags can adjusted/added by agents as a result of some change or action done by the agent. When tags on an agent can only be added/adjusted by other agents a kind of trust and security can be enforced by excluding agents that are tagged as badly behaving agents. Other agents observe this as feedback about the agent carrying the tag and can adjust the tag or add new ones to give further feedback about that agent. If an agent can adjust its own tags then this is a means to give feedback about its own characteristics to other agents. Coordination emerges because agents may discriminate based on tags [51].

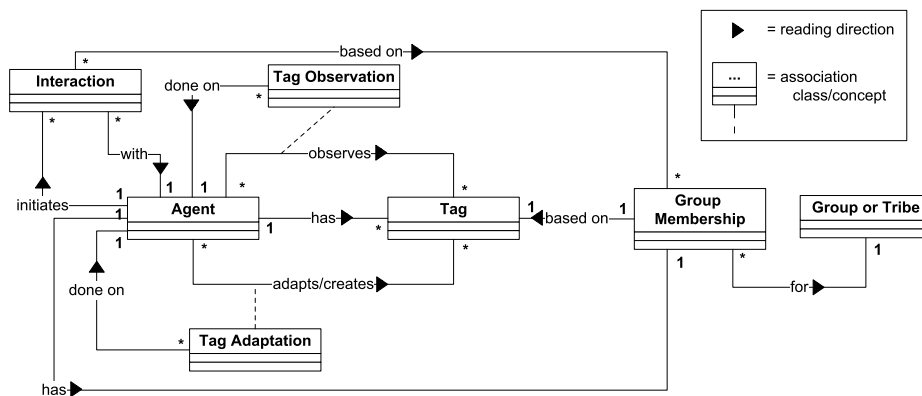


Figure 10. Conceptual model of Tag-based Coordination.

Agents initiate Interactions with other agents based on information obtained by observing each others tags. Assume that agents interact preferentially with those sharing a similar tag. Groups or Tribes are formed around similar tags [7]. Such a tag determines the Group membership of an Agent. Using and adjusting the right tags can serve to achieve desired group cooperations. The key point is that the tags have no direct behavioural implication for the agents that carry them. But through indirect effects, such as the restriction of interaction to those with the same tag value, they can evolve from initially random values into complex ever changing patterns that serve to structure interactions [51].

One major problem that can be solved is the ‘tragedy of commons’. Tags allow to develop coordination mechanisms that discourage selfish behaviour, and encourage altruistic or cooperative behaviour [52]. All individuals benefit if all act altruistically, but each has an incentive to act selfishly. As such tags are a mechanism for promoting cooperation between simple adaptive entities acting in a self-interested way [52].

The key to understanding the tag process is to realise that agents with identical tags can be seen as forming an “interaction group” or “tribe” [52]. The population

can be considered as partitioned into a set of such groups. If a group happens to be entirely composed of cooperating agents then the agents within the group will outperform agents in a group composed entirely of non-cooperating agents. The mechanism is simple. Agents interact in groups (subsets of the population sharing the same tags) [54]. Periodically, if they find another agent who is getting higher utility than themselves they copy them - changing to their group and adopting their strategy. Also, periodically, agents form new groups and/or randomly change their strategies. As such, defectors or selfish agents can do well initially, because they exploit the others in the group, but ultimately all the cooperators leave the group, leaving the selfish alone with nobody to exploit. So a group containing any selfish agents quickly dissolves but those containing only cooperators grow. Given an open system all groups will eventually be invaded by a selfish agent who will exploit and dissolve the group. However, so long as other new cooperative groups are being created then cooperation will persist in the population as a whole. As such, the coordination mechanism effectively emerges an incentive mechanism from the selfish behaviour of the agents. This happens because, although it may do well for a while, a very selfish agent will tend to lose their group members as they find other agents that are members of more cooperative groupings and hence have higher utilities [52].

One can describe this process as a history of one cycle of the mechanism [53]. Very quickly, the population breaks into many disconnected small groups of agents because they quickly form groups with better performing agents. The better performing agents are initially the non-cooperating agents who exploit their groups selfishly. However, this is a non-sustainable strategy because agents in tribes with less exploiters in them do better because they are cooperating as a team. The tribes dominated by selfish agents quickly wither away as agents leave. This emergent property of the birth and death of tribes lays the ground for evolution to operate at the group (tribe) level. As cooperative tribes grow larger they eventually become infected or invaded by a selfish agent. The tribe is quickly destroyed since a selfish agent will exploit the tribe and this will lead to many more agents quickly copying that higher performing node until the tribe is completely selfish and under-performs. As a consequence the group dies because all agents leave it. It is an evolution of the formation, growth, and destruction of tribes. From the simple rules at the individual level an evolutionary process emerges at the tribal level. This process is in constant flux due to mutation and movement, no equilibrium state is attained and no tribe lasts forever. As long as new cooperative tribes are created at least as rapidly as they are destroyed then cooperation can survive.

Similarly *specialisation in skills* can be obtained [54]. Each agent commits to a certain skills. Instead of agents evolving behaviours relating to just cooperation or non-cooperation they evolve discrete skill-types in addition to cooperation giving behaviour. Agents can act selfishly by doing everything and not donating tasks to other more suited agents, or they can donate a task at a certain cost and thus cooperating to get the job done. If agents follow a tag-based evolutionary algorithm then they form groups that contain a diversity of skills within them and sharing of tasks becomes high. The donation rates for donating tasks to other agents are high even when the cost of giving is high to the donating agent.

Parameter Tuning. Some parameters have to be tuned:

- *Probability with which to choose a similar tag:* Of course, agents can be designed to only interact with similarly tagged agents. However, one could introduce a probability parameter that indicates the probability with which to choose a similar tag or not. This can enable another way of exploring.
- *The number of agents involved:* Only having two agents in the system will not enable the tag mechanism to its full extent. But how many agents are needed? This should be tuned with respect to the performance measures.
- *Rate with which agents change groups, i.e. alter their tags versus Rate with which agent change behaviour [52,53]:* What was important here is that the effective mutation rate on the tag should be well over one order of magnitude higher than on the behaviour (cooperation versus selfish behaviour). The tags need to change faster than the behaviour. The mechanism of cooperation is driven by cooperative groups forming more quickly than selfish agents can invade and destroy them. With high mutation on the tag, a cooperative group, as it grows, will tend to spawn more new cooperative groups. As such in large populations the higher utility generating nodes survive and this does not lead to dominance of selfish behaviour because a form of incentive mechanism emerges from the fact that nodes only interact with nodes in the same group (with the same tag).

Infrastructure. The coordination happens directly between agents so no environment-mediated infrastructure is needed. However, it should be possible for agents to carry tags or labels and for other agents to observe them and/or adjust them.

Characteristics.

- *Information Distribution:* Tags are distributed on the agents in the system. As such the information is inherently distributed. This information includes characteristics of the agent carrying the tag and tags can contain any kind of information needed in the problem domain.
- *Feedback Cycle:* Tags are adjusted/added by agents as a result of some change or action done by the agent (e.g. bad behaviour is tagged). Other agents observe this as feedback about the agent carrying the tag and can adjust the tag or add new ones to give further feedback about that agent. If an agent can adjust its own tags then this is a means to give feedback about its own characteristics to other agents.
- *Simple Mechanism - Emergence [52,53,54]:* The agents are rather simple to program. The challenge lays in getting the coordination right. The general goal is to maximise the collective performance of a group while allowing individual agents reasonable levels of autonomy. The macroscopic behaviour emerges from the microscopic interactions based on tags. The tribes are not programmed into the nodes a priori but rather emerge from the tag-based interactions and can produce emergent incentive structures. There is no need to program and test explicit incentive mechanisms for each domain.

An emergent property of the birth and death of tribes lays the ground for evolution to operate at the group (tribe) level. It is an evolution of the formation, growth,

and destruction of tribes. From the simple rules of the mechanism an evolutionary process emerges at the tribal level [53].

- *Scalable [52,53,54]*: The tag mechanism works independently from the number of agents involved. This is preserved when migrating the mechanism from a social science phenomena. It takes no longer to establish cooperation in bigger populations. This is mainly due to its simplicity and inherent decentralisation.
- *Distribution-Decentralisation [52]*: The mechanism requires no central servers or authorities. The mechanism is distributed, each node only has to concern itself with its own interactions.
- *Robustness (against selfish behaviour) [52,53,54]*: Also robustness is preserved when migrating from the social sciences. Tags allow protocols that are robust to node failure, noise or malicious behaviour, such as selfish free riding. In contrast to classical game theoretic approaches, the sociologically inspired approach is more interested in dynamics than equilibrium and in the development of algorithms that can function in noisy environments with incomplete information.
- *Not always an Optimal Equilibrium?:* Compared to a central solution, it is not guaranteed to always be the best way of avoiding selfish behaviour, but it is simple to implement and performs ‘well-enough’ and robust. A central solution is costly and hard to police, does not scale well, is sensitive to noise, and has a high computational overhead.
- *Self-Organising [53]*: A simple selfish protocol that can spontaneously self-organise into internally specialised and/or cooperative groups (or tribes). Agent spontaneously change their behaviour if they come to recognise that they might be able to do better following a different strategy or become member of a different group.
- *Challenge to define a measure of utility [53]*: The tag mechanism for forming groups uses a utility that each agent gains when executing. We assume that some simple measure can be readily calculated and that such measures can be compared between agents. This may not hold in many task domains.
- *Ongoing [53]*: The tag and behaviour mutation process is in constant flux, no equilibrium state is attained and no tribe lasts forever. As long as new cooperative tribes are created at least as rapidly as they are destroyed then cooperation can survive.

7.5 Related Mechanisms/Patterns

Variations

- Tags can have *many representations* [52]. For example, in a network topology we can assume that each node had a fixed capacity of links defining its neighbours (a neighbour list). We can interpret the neighbour list stored in each node as something similar to a tag. If we consider a sparse P2P network in which each node knows some small number of other nodes (neighbours) and those neighbourhoods share a large proportion of other neighbours then in a highly clustered network the same list will be shared by most of the neighbourhood, it defines a group boundary. Interaction with the same tag is to be interpreted as interaction with a node in the neighbourhood. Nodes can then use the performance of neighbouring nodes to decide if they periodically and probabilistically rewire (change their tag) or change their strategy to equal the neighbour.

Other Coordination Mechanisms.

- *Tokens*: Like tags, tokens can encapsulate any information and is held by an agent. However, tags are externally visible properties of those agents and can be adjusted by other agents. Tokens are not externally visible, it is a piece of data held by an agent internally and which can only be adjusted by the agent itself and then passed around to other agents. Also, tags can be copied while only one instance of a specific token can exist.

7.6 Examples/Known Uses

- *Peer-2-Peer File Sharing System*: Massively distributed systems like P2P file sharing systems have the problem that most users only download files rather than sharing them [54]. How can one maintain significant levels of cooperation within such a network composed of selfish and highly autonomous peers [52], i.e. the users and their client software.

The ‘tragedy of commons’ in a P2P application can be described as follows [52]. Users decide on when to download and query the system, can write their own client software, and/or can choose to share nothing or only poor quality files and download many other files, i.e. free-riding. The major problem is to develop mechanisms that discourage selfish behaviour, where peers download files without uploading them, and encourage cooperative behaviour (sharing high quality files). All individuals benefit if all act cooperatively, but each has an incentive to act selfishly.

As indicated in the variations section of this pattern, tags can have many representations. For a peer-to-peer network a tag can be interpreted as the list of neighbouring peers or nodes [52]. As such, interaction with the same tag limits the interaction to the same neighbourhood and performance of neighbouring nodes, measured according to the hit-rate of a node when searching, is used to decide if they periodically and probabilistically rewire (change their tag) or change their strategy to equal the neighbour.

8 Pattern 5: Tokens

Also Known As: key-based [55].

8.1 Context/Applicability

You need to coordinate a large number of autonomous entities in a decentralised way to achieve a common and globally coherent goal. The coordination mechanism has to be robust and flexible in the face of frequent changes and an uncertain environment.

Local estimates of global information are the only possible way to coordinate. As such, decentralised coordination is the only possible alternative. Some locally available information is needed indicating the status of the team and their teammates and allowing to decide on the best behaviour for the collective.

8.2 Problem/Intent

- *Resource Access Control/Allocation/Synchronisation [8]* : How to coordinate the usage of a shared resource so that only a limited number of agents can access it simultaneously? As such, resources are allocated in a safe way and access is synchronised.
- *Information Sharing [8]*: How to share information among teammates in a scalable way and which allows agents to enforce control on how and where the information is shared?
- *Role Allocation [8]*: How to coordinate that all roles in a certain role-based organisation are executed by at least one and at most an unlimited or limited number of agents? As such an organisational structure can be enforced and roles can be passed on when dynamic changes imply a better allocation.

8.3 Forces

- *Centralised vs. Decentralised*: A decentralised solution such as this pattern often requires a huge amount of communication and coordination overhead which a centralised solution has not. A centralised solution can often optimally control the system. However, a centralised solution is often a bottleneck and single point of failure in a very dynamic context. A centralised solution also has a huge amount of communication overhead if the information needed to control the system is intrinsically distributed and a lot of it has to be aggregated to the central node (e.g. large-scale heavy loaded (network) systems). Often, a model of the complete state of a system cannot be obtained at all. As such, decentralised control is the only alternative.
- *Optimality vs. Robustness/Flexibility*: An adaptive approach that has no central means to optimise its efficiency may result in suboptimal solutions. Especially because such a solution relies on local incomplete view about wider system and a local decision can have non-local effects. However, an optimal solution only exists with respect to a static situation, which is never reached in the face of frequent changes. In addition, optimality requires complete information about state of entire system which is impossible. As such, a robust and flexible approach may be preferred to an approach that is optimal but inflexible.

- *Responsibility of Environment vs. Agents*: Coordination often needs complex processing and communication. There is a trade-off to make the agents themselves or the environment in which they are situated responsible for this processing and communication. Making the agents responsible allows agents to explicitly reason about and control how information is distributed but sometimes requires complex algorithms. On the other hand, making the environment responsible allows agents to simply be guided by the results in the environment but the agents are no longer in control of the information distribution.
- *Explore vs. Exploit*: In order to be adaptive the solution has to explore sufficiently compared to only exploiting already known information. Otherwise the approach can get trapped in local optima or never find new targets at all. However, too much exploration may result in an approach that is very inefficient.

8.4 Solution

Inspiration. Most coordination mechanisms are inspired by some natural, biological, physical, or social phenomena. For tokens it is not that clear where the inspiration came from. However, some ideas arise. In a human society there also exist keys that open doors or represent access to certain resources (e.g. computers, etc.). Some keys are also passed among people. Another similar phenomenon is the assignment of roles within companies. A company has resources to hire a certain number of people to take on a certain role in the company. Sometimes this is even one position, e.g. a top manager. In such cases that role is also passed from one person to the next when a top manager leaves the company. Sometimes, even a symbolic key is passed.

Conceptual Description. Tokens are objects that are passed around in a group of agents and which encapsulate anything that needs to be shared by the group, including information, tasks and resources [8]. Agents observe the tokens they are currently holding and all previously incoming and out-going tokens in a kind of historic `PassingPath` [8]. For each type of token there are a limited number of instances and the agent holding a token has exclusive control over whatever is represented by that token. Tokens can be classified into three basic types [8]:

- `Information` tokens represent information to share within a team.
- `Role or Task` tokens represent roles to assign within a team.
- `Resource` tokens represent resources to allocate within a team.

Each token is defined by four elements

- `Type`: the type of coordination it contains (information, role, or resource)
- `Coordination Element`: captures the specific coordination element represented by this token. In the case of an information token, it is the information to be shared. In the case of a resource token, it is a description of the resource to which this token grants exclusive access. Hence, tokens provide a type of access control (*self-protection*). In the case of a role token, it is a description of the task or role for which the acceptor of this token is responsible.

state. Then, by dividing the monolithic joint activity into a set of actions that can be taken by individual agents, we can decentralize the token routing process where distributed agents, in parallel, make independent decisions of where to pass the tokens they currently hold. Therefore, local decision theoretic models to determine when and where to pass tokens are used. When an agent passes a token to another agent, that exchange is used to refine local models of the team. These models are used to determine whether and where to forward any token the agent is currently holding, so as to maximise the expected utility of the team. Informally, agents will try to pass tokens to where they help team performance the most by inferring from their local models which team member will either have use for the information, resource, or task represented by the token or be in the best position to know who will. More specifically, for a token, an agent has a probability model to pass the token to their neighbouring teammates [8]. An agent will route to the teammate with the higher probability. This probability model needs to be updated to optimise the routing of tokens. Initially, agents do not know where to send tokens, but as tokens are received, a model can be developed and better routing decisions made. That is, the model is based on the accumulated information provided by the receipt of previous tokens. For example, when an agent sends a role token to a team mate that has previously rejected a similar role, the team is potentially hurt because this teammate is likely to reject this role too and thus communication bandwidth has been wasted.

- *Previous tokens are feedback for routing of future tokens:* Related to the idea of building local models is to leverage all available information for creating models of the team. Specifically using the movement of one token to inform the movement of other tokens. For example, tokens representing resources useful for a particular task should be passed to the same agent as the token representing that task was. Intuitively, making use of the relationship between tokens, each coordination task becomes more efficient because it focusses its search based on the progress of other coordination tasks. Here the history of received and passed tokens is used.

The effectiveness of the token-based approach depends on how well agents maintain their local models so that tokens are routed to where they lead to the highest gain [8]. The key, in the update algorithm for the probability model of each agent for passing tokens, is to make use of relationships between tokens, which we refer to as relevance. As such the probability to pass a token to a teammate is updated according to the relevance or relationship between the token and previously received tokens from that teammate. For example, receiving a role token from a particular teammate, tells the agent that it is relatively less likely that similar role tokens will be accepted in the part of the network accessible via that neighbour; receiving an information token with information about Pittsburgh tells the agent that some agent in that part of the network must currently be in Pittsburgh. The similarity between tokens used to determine the relevance and relationships between them comes from the coordination element they carry or represent and the calculation depends on the domain knowledge of applications.

Such a coordination mechanism can be used for a number of self-* properties:

- *Resource allocation:* An agent holding a resource token has exclusive access to the resource represented by that token and passes the token to transfer access to

that resource. The resulting movement of tokens implements the coordination by distributing resources.

- *Construction and enforcement of role-based organisations*: a token represents a certain role in the group and by fixing the number of tokens the number of entities in a certain role can be limited. As such an organisational structure can be enforced and roles can be passed on when dynamic changes imply a better allocation.

Parameter Tuning.

- *Control Information*: If control information in the form of thresholds or maximum number of hops is used then these are also parameters to tune.
- *Probability to choose best destination for token*: To tune how much agents exploit information from previously received tokens to decide to which teammate to pass a token. A parameter can be used that indicates how certain the teammate with the highest estimated need for the token is chosen.

Infrastructure. No special infrastructure is needed. Only the possibility to pass around data objects (tokens) through communication channels.

Characteristics.

- *Low communication overhead [8]*: The resulting movement of tokens implements the coordination by distributing information, resources, and tasks with low communication overhead because tokens are only passed between neighbouring teammates (one-on-one) and not to all teammates every time.
- *Scalable to Large Teams [8,55,56]*: There is effective coordination independent of the size of the team. The communication overhead does not increase significantly when the size of the team increases.
- *Decentralised-Distributed [8]*: No central control is needed. The passing of tokens happens completely decentralised and distributed in the team. As a consequence, especially in large teams, knowing the complete team state is infeasible. Agents must make token coordination decisions based on a more limited view of the team, i.e. tokens they are currently holding and all previously incoming and out-going tokens. The resulting approach allows fast, efficient routing decisions without requiring accurate knowledge of the complete state.
- *Optimality vs. robustness/flexibility [8]*: Of course, such a decentralised solution performs less optimal than a central solution optimising with complete knowledge. However, the coordination is very robust and flexible in the face of frequent changes which is inherent to the problems where it is used.
- *Emergence - Feedback [8]*: Token based coordination is a process by which agents attempt to maximize the overall global team reward by moving tokens around the team locally, i.e. coordination emerges. Each token received by an agent is used as feedback to improve routing of other future tokens leading to dramatic global performance improvements.

- *Coordination is responsibility of agents*: The agents themselves are responsible and can control the distribution of information with tokens. More specifically, the tokens include extra control information usable to decide to where the token is passed by agents. In contrast, information sharing with for example co-fields enforces that the environment is responsible for distribution and agents no longer control what information is distributed where.
- *Explore vs. Exploit*: This trade-off can be tuned in the token-based approach by increasing or decreasing the certainty with which a token is passed to the teammate with the highest probability of needing the token (calculated from information out of the history of received tokens).
- *Information Distribution*: Tokens deliver information on roles and tasks to be performed, locks acquired on resources, and other information embedded into a token.

8.5 Related Mechanisms/Patterns

Other Coordination Mechanisms

- *Tags*: Like tokens, tags can encapsulate any information and is held by an agent. However, tags are externally visible properties of those agents and can be adjusted by other agents. Tokens are not externally visible, it is a piece of data held by an agent internally and which can only be adjusted by the agent itself and then passed around to other agents. Also, tags can be copied while only one or a limited number of instances of a specific token can exist.

8.6 Examples/Known Uses

- *Unmanned Aerial Vehicles (UAVs) searching a hostile area [8]*: The following types of tokens were used:
 - **Information Tokens**: Detections of found targets are passed around the team as pieces of information.
 - **Resource Tokens**: Airspace over the hostile area was divided into 50 regions. Each of these regions were duplicated in three resource tokens allowing a maximum of three UAVs to simultaneously access that region.
 - **Role Tokens**: Surveillance plan instances, each with four independent preconditions, were given to the team. After a plan was initiated, tokens for the four roles needed to realize the plan were circulated through the team network. To accept a role an agent must be close to the location the role requires and have access to resource tokens for airspace at that role location.

This resulted in a coordinated behaviour that performed well, used a low communication bandwidth, and is scalable to larger teams because the message overload did not increase with larger teams.
- *Other uses [8]*: Large scale coordination in the GPGP/TAEMS framework was demonstrated using a token-based algorithm [55]. The effectiveness of large-scale token-based coordination has also been demonstrated in the Machinetta proxy architecture [58] for task allocation [57] and information sharing [56] and others [59].

9 Case Study: A Packet Delivery Service

This section considers the design of a packet delivery service application to illustrate the use of decentralised coordination patterns. First the requirements are described and then a design is proposed using the patterns.

Problem Statement and Requirements. A packet delivery service [60] allows customers to submit an order to come and pick up a packet and transport it to a given destination or delivery location (see figure 12). At each moment in time, new pickup and delivery locations can be added to the system by customers. A fleet of trucks is available which has to self-organise efficiently to accommodate the current request for transport. As such, the orders of customers form a dynamic overlay network of pickup and delivery locations between which trucks have to move routing themselves through a street map. So basically there are two main requirements for this problem:

- Dispatching: every new order has to be assigned to a truck that will be responsible for handling the requested transport.
- Routing: trucks have to be adaptively routed through a street map in order to reach new pickup locations for orders to which they were assigned and to reach delivery locations for orders already inside the truck.

These requirements have to be achieved in the face of frequent changes: new orders arriving at any moment, changes to the delivery location of existing orders, congestion and obstacles on streets, trucks failing, etc. On the other hand there are a number of timing constraints that have to be reached: pickup time-window in which the pickup should occur, delivery time-window, regular breaks for drivers to rest, etc. This is a highly dynamic problem in which the information needed to decide how to route or dispatch is inherently decentralised over a number of trucks, customers, streets, etc. As such a self-organising emergent solution is promising.

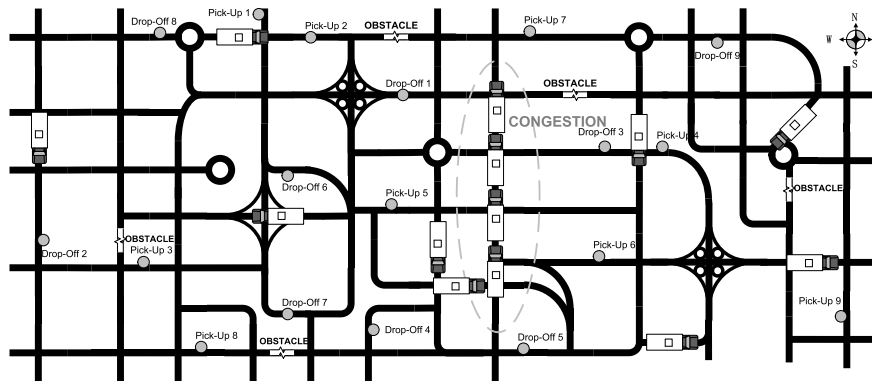


Figure 12. The Packet-Delivery Problem.

Design with Decentralised Coordination Patterns. As mentioned earlier, the problem-solving power resides in the interaction between agents. Therefore for each of the requirements that have to be coordinated one has to discover which information is needed to take the appropriate decisions and actions. And especially, which decentralised coordination mechanism allows to exchange that kind of information and to coordinate the agents to achieve the requirements.

Consider the *dispatching requirement* for which new orders have to be assigned as a task to available trucks. Information is needed to decide which truck is chosen, such as the distance of the truck to the pickup location or its estimated arrival time, the trucks available with enough room to carry the packet, and the estimated delivery times of trucks. The best truck at that moment should be allocated. To solve this problem systematically, the ‘Problem/Solution Summary’ described in Table 1 is used. The engineer has to find a problem description matching the dispatching problem. Some kind of allocation of resources is required. The resource is the room available in trucks that has to be allocated to an order. Table 1 states that a Market-based Coordination mechanism may be suitable. Therefore, the consumers and suppliers of the resource market have to be determined. For example, consider the trucks as the suppliers of available transportation room, and order agents, representing orders, are the consumers of that room. Another important aspect of markets is the instantiation of the price mechanism. According to the pattern, a number of price values are involved such as limit prices under which the room is not bought or sold, prices offered in individual bids, and the market price. These prices should depend on information important for making the allocation decisions because in a market limit prices, offered prices, and market price determine what is allocated to whom. As such, for the order dispatching a truck could have a limit price that increases with the distance or estimated travel time to reach the pickup, increases with the estimated delivery time, decreases with the amount of available room in the truck, etc. The order’s limit price can depend on the wanted delivery and pickup time constraints so that an order willing to pay a high price is willing to wait longer before pickup and/or doesn’t require a short delivery time. A market is started when an order is submitted by a customer and stops only when it has been assigned to a truck. During the market execution trucks will offer to supply room at a certain price as far as possible above their limit price to maximise their profit (i.e. for example, the higher the price payed, the more time the truck has to deliver the order). Orders will bid to buy the room at certain prices that are as far as possible below their limit price in order to get the best deal (i.e. for example, the lower the price, the quicker the delivery is done). The market mechanism of demand and supply then allocates the trucks to the orders in such a way that approximates a globally efficient optimum at that moment.

The second requirement was to actually *route the trucks through the street map*. The problem to be solved is a routing or spatial movement problem for which there are two possible coordination patterns according to table 1: digital pheromones or gradient fields. To decide which one is the most promising the patterns need to be studied in more detail, i.e. which characteristics match the needs for routing trucks, etc. Assume that the gradient field approach is the most suitable mainly because the digital pheromone approach requires that trucks actively search for delivery locations, orders, etc. On the other hand, the gradients are automatically propagated by the environment and trucks

only have to follow their coordination gradient to be routed through the street map efficiently. As such new orders are immediately found by trucks which is an important requirement. Without giving a more detailed solution the main idea is using different types of gradients:

- *Delivery Location Gradients*: each delivery location emits a gradient as soon as the order is assigned to a truck. The truck then simply follows that gradient.
- *Pickup Location Gradients*: each pickup location emits a gradient to signal the presence of a new order. The truck assigned to the order simply follows the gradient to reach the pickup.
- *Market Communication Gradients*: to facilitate direct interaction and negotiation needed for markets between trucks and orders at pickup locations each can emit a market gradient. That gradient is used to actively send bids to buy room, offers to sell room, payments, etc. These messages can route themselves by simply following the right gradient. Each truck and order have their own gradient.

Note that a requirements was to cope with dynamics such as congestion and obstacles. The gradients used above all take into account these changes in their propagation rules. For example, the propagation occurs slower, not at all, or with a higher increase in strength through congested streets. Similar for other dynamics.

Once the coordination mechanisms are chosen, the pattern description offers a guide to actually apply them: different variants can be considered, the parameters to tune are known, guidelines are available, etc. In particular, a gradient solution requires an environment capable of storing and propagating gradients, i.e. an infrastructure. Such an environment is not available or too expensive on a real street network. Alternatively, a decentralised solution which is not distributed on the street network can be used. As such a server emulates the street network as a graph environment in which gradients propagate and on which truck agents move in sync with and actively coordinate the movement of the real trucks. Changes such as information on congestion and obstacles is updated in that emulated environment in a decentralised manner by directly linking the real world trucks with the agents. Customers submit their order to this server. A new order results in an order agent appearing at the corresponding pickup location in the emulated environment. The advantages of a self-organising solution are preserved because the solution is constantly adapting to changes without a central controlling entity and is still robust to truck failure. Of course, a failure of the server emulating the environment would break the system. However, making the server failure-safe remains cheaper than embedding a gradient infrastructure in a real street network.

10 Conclusion and Future Work

To systematically design a self-organising emergent solution, guidance on which decentralised coordination mechanisms to use is essential because the problem-solving power resides in the coordination process. Decentralised coordination mechanisms can be described as design patterns, similar to patterns used in mainstream software engineering. Such a clear and structured description format helps in making the engineering process more systematic for two reasons.

- Firstly, pattern descriptions allow to *directly find a solution based on the problem*.
- Secondly, each *pattern is a consolidation of best practice* to use it, which *systematically guides engineers* in applying the coordination mechanisms.

However, the patterns in this paper have a conceptual focus. More work is needed on patterns for designing coordination mechanisms at the class or implementation level. Even for the conceptual patterns given, more structure is possible by for example identifying the participants and interactions and represent this structurally in UML interaction diagrams. Also, many other coordination mechanisms should be captured as design patterns to easily compare and choose between them when engineering a self-organising emergent solution.

Acknowledgements. This work is supported by the K.U.Leuven research council as part of the concerted research action on Autonomic Computing for Decentralised Production Systems (project 3E040752).

References

1. Herrmann, K., Mhl, G., Geihs, K.: Self-Management: The Solution to Complexity or Just Another Problem? IEEE Distributed Systems Online 6(1) (2005)
2. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer Magazine 36(1) (2003) 41–50
3. De Wolf, T., Holvoet, T.: Emergence versus self-organisation: different concepts but promising when combined. In: Engineering Self Organising Systems: Methodologies and Applications. Volume 3464 of Lecture Notes in Computer Science., Springer Verlag (2005) 1–15
4. Brueckner, S.: Return From The Ant - Synthetic Ecosystems For Manufacturing Control. PhD thesis, Humboldt-Universitt, Berlin (2000)
5. Mamei, M., Zambonelli, F., Leonardi, L.: Co-fields: A physically inspired approach to motion coordination. IEEE Pervasive Computing 3(2) (2004)
6. S. H. Clearwater, E.: Market-Based Control: A Paradigm for Distributed Resource Allocation. World Scientific, Signapore (1996)
7. Hales, D.: Choose your tribe! - evolution at the next level in a peer-to-peer network. In: Proc. of the 3rd Workshop on Engineering Self-Organising Applications (EOSA 2005). (2005)
8. Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., Sycara, K.: An integrated token-based algorithm for scalable coordination. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS), Utrecht, NL (2005) 407–414
9. Mamei, M., Zambonelli, F., Leonardi, L.: Distributed motion coordination with co-fields: A case study in urban traffic management. In: Proc. of the The 6th Int. Symp. on Autonomous Decentralized Systems (ISADS'03), Washington, DC, USA, IEEE CS (2003) page 63

10. Mamei, M., Vasirani, M., Zambonelli, F.: Experiments of morphogenesis in swarms of simple mobile robots. *Applied Artificial Intelligence* **18**(9-10) (2004) 903–919
11. Mamei, M., Zambonelli, F.: Theory and practice of field-based motion coordination in multi-agent systems. *J. Appl. Artif. Intell.* **19** (2005) to be published.
12. Mamei, M., Zambonelli, F.: Motion coordination in the quake 3 area environment: A field-based approach. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: *Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, July 19, 2004, Revised Selected Papers*. Volume 3374 of *Lecture Notes in Computer Science.*, Springer Verlag (2005) 264
13. Rimon, E., Kodischek, D.E.: Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation* **8**(5) (1992) 501–518
14. Ygge, F., Akkermans, H.: Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research* **11** (1999) 301–333
15. Cliff, D., Bruten, J.: Simple bargaining agents for decentralized market-based control. Technical Report HPL-98-17, HP Labs, Bristol, UK (1998)
16. Gibney, M.A., Jennings, N.R., Vriend, N.J., Griffiths, J.M.: Market-based call routing in telecommunications networks using adaptive pricing and real bidding. In: *IATA '99: Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications*, London, UK, Springer-Verlag (1999) 46–61
17. Guenther, O., Hogg, T., Huberman, B.: Power markets for controlling smart matter. *Computing in Economics and Finance 1997* 62, Society for Computational Economics (1997)
18. Lynch, J.P., Law, K.H.: Decentralized control techniques for large-scale civil structural systems. In: *Proc. of the 20th Int. Modal Analysis Conference (IMAC XX)*. (2002)
19. Parunak, H.V.D., Baker, A.D., Clark, S.J.: The aaria agent architecture: From manufacturing requirements to agent-based system design. *Integrated Computer-Aided Engineering* **8**(1) (2001) 45–58
20. Etzioni, O.: Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web. In: *Proc. of the 13th Int. Conf. on Artificial Intelligence*. (1996)
21. Weyns, D., Helleboogh, A., Steegmans, E., De Wolf, T., Mertens, K., Bouck, N., Holvoet, T.: Agents are not part of the problem, agents can solve the problem. In: *Proc. of the OOPSLA Workshop on Agent-Oriented Methodologies*. (2004)
22. Wooldridge, M., Jennings, N.R.: Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing* **3**(3) (1999) 20–27
23. De Wolf, T., Holvoet, T.: Towards a methodology for engineering self-organising emergent systems. In Czap, H., Unland, R., Branki, C., Tianfield, H., eds.: *Self-Organization and Autonomic Informatics (I)*. Volume 135 of *Front. in Artif. Intell. and App.*, IOS Press (2005)
24. Jacobson, I., Booch, G., Rumbaugh, J.: *The unified software development process*. Addison Wesley (1999)
25. Larman, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd edn. Prentice Hall (2005)
26. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison-Wesley (1995)
27. Meszaros, G., Doble, J.: Metapatterns: A pattern language for pattern writing. In: *The 3rd Pattern Languages of Programming conference*, Monticello, Illinois, USA (1996)
28. Caro, G.D., Dorigo, M.: Two ant colony algorithms for best-effort routing in datagram networks. In: *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*, IASTED/ACTA Press (1998) 541–546
29. Parunak, H.V.D., Brueckner, S.A., Sauter, J.: Digital pheromones for coordination of unmanned vehicles. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: *Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, July 19, 2004,*

- Revised Selected Papers. Volume 3374 of Lecture Notes in Computer Science. Springer-Verlag GmbH (2005) 246–263
30. Panait, L., Luke, S.: A pheromone-based utility model for collaborative foraging. In: Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04). Volume 1. (2004) 36–43
 31. Weinstein, P., Parunak, H.V., Chiusano, P., Brueckner, S.: Agents swarming in semantic spaces to corroborate hypotheses. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, IEEE Computer Society (2004) 1488–1489
 32. Valckenaers, P., Kollingbaum, M., Van Brussel, H., Bochmann, O., Zamfirescu, C.: The design of multi-agent coordination and control systems using stigmergy. In: Proceedings of the IWES'01 Conference, Bled, Slovenia (2001)
 33. Brueckner, S.A., Parunak, H.V.D.: Swarming distributed pattern detection and classification. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, July 19, 2004, Revised Selected Papers. Volume 3374 of Lecture Notes in Computer Science. Springer Verlag (2005) 232–245
 34. Sauter, J.A., Matthews, R., Parunak, H.V.D., Brueckner, S.A.: Performance of digital pheromones for swarming vehicle control. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press (2005) 903–910
 35. De Wolf, T., Holvoet, T.: A Taxonomy for Self-* Properties in Decentralised Autonomic Computing. In: Autonomic Computing: Concepts, Infrastructure, and Applications. CRC Press (to appear in 2006)
 36. Brueckner, S., Parunak, H.: Multiple pheromones for improved guidance. In: Proceedings of Symposium on Advanced Enterprise Control. (2000)
 37. Dorigo, M., Maniezzo, V., Coloni, A.: Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B* **26**(1) (1996) 29–41
 38. Gregorio, J.: Bitworking: Stigmergy and the world-wide web. online article on [http : //bitworking.org/news/Stigmergy](http://bitworking.org/news/Stigmergy) (2002)
 39. Parunak, H.V.D., Brueckner, S.A.: Stigmergic Learning for Self-Organizing Mobile Ad-Hoc Networks (MANET's). In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04). Volume 3. (2004) 1324–1325
 40. Van Dyke Parunak, H., Brueckner, S., Matthews, R., Sauter, J.: Pheromone learning for self-organizing agents. *IEEE Transactions on Systems, Man and Cybernetics, Part A* **35**(3) (2005) 316– 326
 41. Cabri, G., Leonardi, L., Zambonelli, F.: Engineering mobile agent applications via context-dependent coordination. *IEEE Transactions on Software Engineering* **28**(11) (2002) 1039–1055
 42. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Thomas F. Knight, J., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. *Commun. ACM* **43**(5) (2000) 74–82
 43. Howard, A., Mataric, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots* **13**(2) (2002) 113–126
 44. Leonardi, L., Mamei, M., Zambonelli, F.: Co-fields: Towards a unifying model for swarm intelligence. Technical Report DISMI-UNIMO-3-2002, University of Modena and Reggio Emilia, Modena, Italy (2002)
 45. Mamei, M., Zambonelli, F., Leonardi, L.: A physically grounded approach to coordinte movements in a team. In: Proceedings of First International Workshop on Mobile Teamwork (at ICDCS), IEEE CS Press (2002)

46. Varian, H.: *Intermediate Microeconomics*. W.W.Norton, New York, USA (1999)
47. Akkermans, H., Schreinemakers, J., Kok, K.: Emergence of control in a large-scale society of economic physical agents. In: *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Washington, DC, USA, IEEE Computer Society (2004) 1232–1234
48. Jackson, W.B., Mochon, C., Van Schuylenbergh, K., Biegelsen, D.K., Fromherz, M.P.J., Hogg, T., Berlin, A.A.: Distributed allocation using analog market wire computation and communication. In: *Proceedings of the 7th Mechatronics Forum International Conference*, Atlanta, GA (2000)
49. Parunak, H.V.D.: A survey of environments and mechanisms for human-human stigmergy. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: *Post-proceedings of E4MAS 2005 workshop: Environments for Multi-Agent Systems*. Volume 3830 of *Lecture Notes in Computer Science*. Springer Verlag (to appear in 2006)
50. Axtell, R., Epstein, J.: *Distributed Computation of Economic Equilibria via Bilateral Exchange*. Brookings Institution, Washington DC (1997)
51. Holland, J.: The effect of labels (tags) on social interactions. Technical Report SFI Working Paper 93-10-064, Santa Fe Institute, Santa Fe, NM (1993)
52. Hales, D., Edmonds, B.: Applying a socially inspired technique (tags) to improve cooperation in p2p networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A* **35**(3) (2005) 385–395
53. Hales, D.: Choose your tribe! - evolution at the next level in a peer-to-peer network. Technical Report UBLCS-2005-13, Department of Computer Science, University of Bologna (2005)
54. Hales, D.: Engineering with sociological metaphors: Examples and prospects. In: *Proceedings of the AISB2005 Symposium - Engineering with Social Metaphors*. (2005)
55. Wagner, T., Guralnik, V., Phelps, J.: A key-based coordination algorithm for dynamic readiness and repair service coordination. In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*. (2003)
56. Xu, Y., Lewis, M., Sycara, K., Scerri, P.: Information sharing in very large teams. In: *Proceedings of the AAMAS'04 workshop on Challenges in Coordination of Large-Scale MultiAgent Systems*. (2004)
57. Scerri, P., Farinelli, A., Okamoto, S., Tambe, M.: Allocating tasks in extreme teams. In: *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. (2005)
58. Scerri, P., Pynadath, D., Johnson, L., Rosenbloom, P., Shurr, N., Si, M., Tambe, M.: A prototype infrastructure for distributed robot-agent-person teams. In: *Proceedings of the second international joint conference on autonomous agents and multiagent systems*. (2003)
59. Scerri, P., Xu, Y., Liao, E., Lai, J., Sycara, K.: Scaling teamwork to very large teams. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. (2004)
60. Savelsbergh, M.W.P., Sol, M.: The general pickup and delivery problem. *Transportation Science* **29** (1995) 1729