Foundations for a Mathematical Model of the Global Brain: architecture, components, and specifications

Francis Heylighen, Evo Busseniers, Viktoras Veitas, Clément Vidal & David. R. Weinbaum
Global Brain Institute, Vrije Universiteit Brussel

Abstract:
The global brain can be defined as the distributed intelligence emerging from the network of all people and machines on this planet, as connected via the Internet. The present paper proposes the foundations for a mathematical model of the self-organization of such a network towards increasing intelligence. The assumption is that the network becomes more efficient in routing the right information to the right people, so that problems and opportunities can be addressed more efficiently by coordinating the actions of many people. The network develops by the creation and strengthening of useful connections, and the weakening and eventual elimination of counterproductive connections.

People are modeled as agents that try to maximize their benefit by processing the challenges (problems and/or opportunities) they encounter. Challenges propagate from agent to agent across a weighted, directed network, which represents the social connections between these agents. A challenge is defined as the difference between the actual situation encountered by the agent, and that agent's need (i.e. the agent's desired situation). Challenges, computed as the difference between a situation and a need, have a variable state represented as a sparse vector, i.e. a list of real numbers most of which are 0. Negative numbers represent problems or deficiencies, positive numbers opportunities or resources. Agents deal with challenges by multiplying the corresponding vector by their processing matrix, thus producing a new situation vector. An agent's processing matrix represents its skill in relaxing the different components of the challenge towards 0 (i.e. reducing the difference between situation and need). The degree of relaxation defines the amount of benefit extracted by the agent. Agents receive challenges either randomly from the challenge generator, or selectively from other agents, after these agents have extracted some benefit. Challenges are transmitted along the links between agents, with a probability dependent on the strength of the link. This strength evolves through a reinforcement-learning rule: the more benefit sending and receiving agents extract, the stronger their link becomes. In this way, the network self-organizes in a way similar to a neural network, while increasing the total amount of benefit extracted by all agents collectively.

The intention of the simulation is to explore the space of parameters and propagation mechanisms in order to find the configurations that maximize this collective benefit extraction, which we define as the distributed intelligence of the network. Some of the parameters and mechanisms to be explored are the following: the capacity of the challenge queue (buffer memory) from which an agent selects the most promising challenges for processing; the criteria (challenge intensity, assumed processing skill, trust in the sender…) that an agent uses to select the most promising challenge; the relative proportion of rival and non-rival components of the challenge vector, where a rival component represent a material resource whose value is consumed by the processing, while a non-rival one represents an informational resource that maintains it value; the agent "IQ", defined as the ability of the agent's processing matrix to reduce the challenge intensity; the agent "mood", determined by the sequence of its most recent successes and failures in benefit extraction, which affects its willingness to take risks; and an agent's "preference vector", defined by its track record in accepting or rejecting challenges. In a later stage, we also envisage to include market mechanisms, in which agents "pay" others with raw benefit in order to receive promising challenges, and reputation mechanisms, in which the reliability of an agent with which the present agent has no experience yet is estimated on the basis of its relations with other agents.

The basic model has been tested through a prototype implementation in Matlab. This confirmed that distributed intelligence effectively increases under the given assumptions. We are now developing a more detailed and scalable simulation model, which can potentially include millions of agents and links, using distributed graph databases and graph traversal algorithms. This will allow us to explore the effects of a large number of variations of the parameter values and mechanisms. Because of the modular architecture of the implementation, the effect of parameters or mechanisms can be studied in isolation or in various combinations. We hope that this will allow us to elucidate the influence of variables such as network topology, individual intelligence and type of social interaction on the emergence of distributed intelligence on the Internet.

We plan to eventually compare the results of our simulations with empirical data, such as the propagation of Twitter, email or Facebook messages across an existing social network. For this we can use Latent Semantic Analysis (LSA) techniques to convert the text of the messages to the kind of vectors used in the model to represent challenges. If the model is successful, it should be able to predict such propagation much better than chance.

1. Introduction

The *global brain* can be defined as the distributed intelligence emerging from all people and computers on this planet, as communicating via the Internet (Bernstein, Klein, & Malone, 2012; Goertzel, 2002; Heylighen, 2011a). The metaphor of "brain" for this intelligent network is apt, given that, just like the human brain, this system processes information in order to exploit opportunities, solve problems, learn new knowledge, decide about strategies, and take action. This brain is "global" in the sense that it spans the globe, and that it functions in a sense like a nervous system for the Earth, connecting and coordinating all decision-making agents on this planet.

When we say that this intelligence "emerges", we use the word in both its static and dynamic meanings. Statically, an emergent property is one that is situated at the level of the whole system and that cannot be reduced to the properties of its parts (Anderson, 1972; Heylighen, 1991). This is what underlies the idea of collective intelligence or "wisdom of crowds": a group often can solve problems that none of its individual components can solve; the whole is smarter than the aggregate (sum) of its parts. Emergent properties are typically the result of *self-organization*(Corning, 1995; Heylighen, 2001, 2008): the system rearranges itself so as to develop a more coherent, coordinated structure, capable of new functions. The human brain is a classic example of a self-organizing system: it exhibits emergent properties—such as intelligence, goal-directedness or consciousness—that cannot be found in its components—neurons and synapses.

Dynamically, self-organization is a process that takes time, and that typically needs to explore a variety of states before it manages to settle into a stable, adapted configuration (Heylighen, 2001). In that sense, the intelligence of the global brain is still in the process of emerging: it has far from settled, and is as yet still rudimentary in its capabilities.

While the concepts of emergence, brain and intelligence seem appropriate to describe the coordinating function exhibited by the Internet, these concepts are as yet not very well understood. Therefore, they are typically used in vague, metaphorical ways, often obscuring the issue more than clarifying it. The way science avoids such ambiguity is through formalization (Heylighen, 1999): expressing concepts and principles as explicitly as possible, preferably in mathematical form, so that they can be checked without ambiguity as to what is being checked. Self-organization is commonly modeled with the help of dynamical systems, a mathematical representation that allows us to compute the trajectory of a system through its state space, from its initial, transient state to a stable "attractor" regime.

The hypothesis of the emerging global brain entails that movement along this trajectory would be accompanied by some measurable increase in intelligence. This intelligence should be *distributed* (Heylighen, Heath, & Van Overwalle, 2004; Hutchins, 2000; Rumelhart & McClelland, 1986), in the sense that it is not localized in one or a few components, but that it emerges from the network of interactions between all the components.

Complex systems are typically modeled as sets of interacting agents (Bonabeau, 2002), where an agent is an elementary component that applies given rules to react to specific conditions with specific actions. Thus, each agent behaves like a simple mathematical function or automaton. Emergence arises because the action of an agent normally affects the actions of the agents in its neighborhood, which in turn affect the agents in their neighborhood. Thus, simple, local changes can propagate across an ever-widening neighborhood, eventually determining the global dynamics of the system. Because of the non-linearity caused by the feedbacks between agents, this dynamics is normally unpredictable without a detailed follow-up of all the local events, which are affected by random fluctuations. Therefore, complex adaptive systems are typically modeled by means of computer simulations of multi-agent systems (Miller & Page, 2007; Woolridge & Wooldridge, 2001), in which many different random configurations are tried out, in the hope of finding statistically recurrent patterns.

The present paper wishes to lay the foundations for such a model of the global brain (GB), hoping to explore how its distributed intelligence would emerge and self-organize. The model aims to be as realistic as possible while remaining simple enough for conceptual and mathematical clarity, so that the different components of the model can be understood and investigated both formally and intuitively.

2. Conceptual foundations of the model

Our model of the GB is based on the paradigm of *challenge propagation*, which has been described elsewhere (Heylighen, 2011c, 2012a, 2012b; Weinbaum, 2012). Therefore, we will not go into details here, but merely summarize the main ideas.

We conceive individual people and software systems as *agents* that act on *challenges*. A challenge is a situation that incites action, by promising a benefit if the challenge is adequately dealt with, or threatening with a penalty if the challenge is ignored. Agents have limited abilities or skills. Therefore, they cannot optimally tackle all aspects of a challenge. Moreover, the same situation can present different benefits or threats to different agents, because these agents have different needs. Therefore, agents will in general pass on challenges to their collaborators or "friends", i.e. their trusted connections, so that these friends too may be able to extract some of the remaining benefit, or tackle some of the unsolved problems. Thus, challenges propagate from agent to agent, while undergoing benefit-extracting processing, until they no longer offer any significant benefit.

For a concrete illustration of the challenge propagation model, imagine a typical social media network, such as Facebook or LinkedIn. Each member of the network is linked to a group of friends or contacts. When members have a question, an observation, or something interesting to share, they post it on their personal page in the network, where some or all of their friends can see it. These friends can react on the post, e.g. by answering the question, commenting on the suggestion, checking the link, or reposting the information on their own personal page, where it will be seen by *their* friends. The most interesting observations may thus be posted and reposted to increasingly remote groups of friends of friends, spreading ever more widely across the network. Each post is a challenge for the readers of the social media page—most frequently inciting them to enjoy, use or otherwise extract benefit from what was posted, less commonly to help the author of the post to solve some problem, or to attract their attention to a problem that also concerns them.

The connections between agents define a social network. This is a directed, weighted graph with a variable topology. Connections can be created, strengthened, weakened or cut, depending on their usefulness: if the agent receiving a challenge extracts benefit from it, it will tend to strengthen its link with the sending agent, so as to receive more of this kind of challenges in the future; if not, it will tend to reduce the strength of the link. As such, the social network as a whole "learns" the most effective pattern of connections, in a way similar to the learning dynamics used in connectionist, "neural" networks (Heylighen & Bollen, 2002; McLeod, Plunkett, & Rolls, 1998; Van Overwalle & Heylighen, 2006).

Our hypothesis is that this learning process will increase the intelligence of the distributed cognitive system formed by all agents and their connections. We assume that agents are individually intelligent, in the sense that they can to some degree solve the problems posed by benefit extraction from challenges. In principle, by efficiently dividing the labor, and coordinating their efforts, agents working together should be able to extract more benefits than when working separately. The surplus of problems solved or benefit extracted by the social system, as compared to the scattered individuals, constitutes the distributed intelligence of the system.

According to our basic hypothesis, this surplus should increase over time as the learning mechanism optimizes the structure of the network, so that the right challenges are delegated at the right time to the right agents. But both the surplus and its growth will depend on a variety of parameters characterizing the network, the agents, and their (inter)actions. These dependencies are what a realistic simulation model should allow us to map. Once we get a better understanding of which parameters have what effect, we may be able to make recommendations about optimizing the system, i.e. suggesting an efficient strategy to make the network more intelligent.

3. Architecture of the model

3.1 Main components

In summary, the basic components of the model are:

- a set of potential challenges $C = \{c\}$, each characterized by its challenge vector $\mathbf{c} = (c_1, c_2, \dots c_N) \in \mathbb{R}^N$ that represents the present state of the challenge
- a set $A = \{a_i\}$ of agents, each characterized by its own need vector $\mathbf{n}(a_i) \in \mathbb{R}^N$, and its processing matrix $\mathbf{M}(a_i) = (m_{jk}(a_i))$ that represents its skills in dealing with challenges
- an algorithm (the challenge source or generator) that creates a stochastic distribution of challenges and delivers them to random agents
- a network of directed links between agents: $l_{ij} = (a_i, a_j)$, with weights: $w(l_{ij}) = w_{ij} \in [0, 1]$
- a challenge queue for each agent, $Q(a) = (c_1, \dots c_q)$, holding a limited number q of received challenges for further processing
- a system of rules that determines how each agent selects, processes, extracts benefit and propagates the challenges in its queue
- a learning algorithm that determines the creation, deletion and reinforcement of links, and thus the adaptive topology of the network
- a measure D for the distributed intelligence of the network, based on the total benefit extracted

3.2 Flow of Activity

The general activity in the model system can be summarized as follows.

Situation vectors (potential challenges) are generated by an algorithm that produces a power law distribution around 0 for each scalar component of the vector, with most entries equal to zero (i.e. the vectors are sparse). These continuously generated vectors are assigned to random agents, who add them to their queue. Each agent scans the vectors in its queue, transforming them into actual challenge vectors by subtracting them from its need vector. It then calculates the intensity (importance) of the challenge. On the basis of this intensity, the agent selects the most important challenge for processing.

Processing means transforming the vector into a new vector using a linear transformation, determined by the agent's processing matrix. The processed vector normally has a reduced ("relaxed") intensity, depending on the processing ability of the agent. The degree to which the intensity has been reduced defines the benefit that the agent has extracted from the challenge. The agent then decides whether or not to propagate the relaxed challenge via its outgoing links, depending in part on the remaining intensity.

Propagated challenges are added to the queue of the receiving agents, where they are taken into account with a probability dependent on the weight of the link. If such a received challenge is processed by the agent, the benefit extracted is used to increase the link weight by an amount proportional to that benefit (which may be negative, in which case the link strength decreases).

Once the limited capacity of a queue has been fully used, newly incoming challenges can only be added by removing older challenges from the queue. The selection criterion is intensity: the least intense challenge is deleted to make space for a more intense one. New challenges less intense than the least intense one in the queue are immediately deleted (and thus effectively ignored).

Deleted challenges are relegated to a challenge “sink”, where they are stored for later inspection by the researchers.

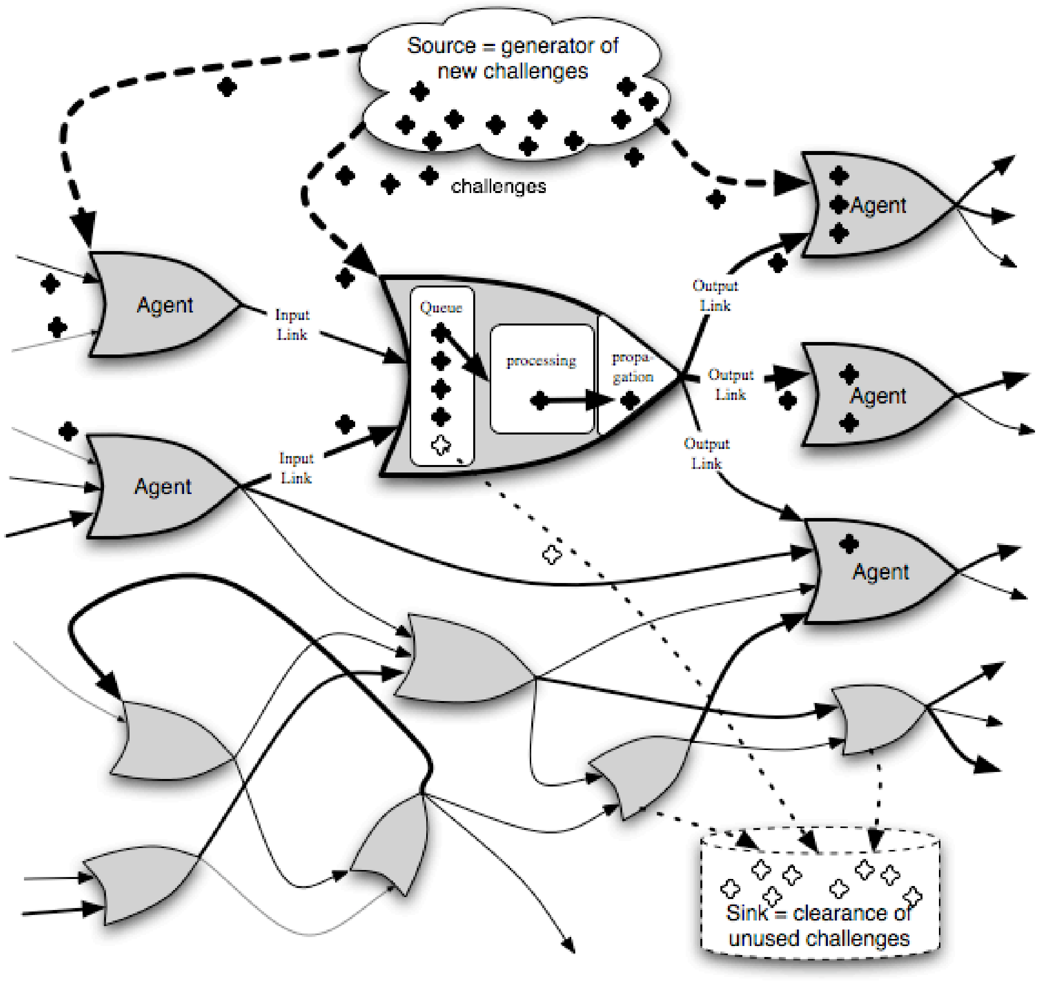
Source = generator of new challenges
challenges
Agent
Input Link
Queue
processing
propa gation
Output Link
Sink = clearance of unused challenges

Figure 1: a sketch of the model’s overall architecture.

Agents connected by links form a weighted, directed network. Challenges (denoted by “+” signs) originate in the source and are sent to individual agents, where they are temporarily stored in the agent’s queue. The agent then selects the most important challenge to process and (potentially) propagate further across outgoing links to “friends” of the agent. A propagated challenge is added to the receiving agent’s queue where it competes for the chance of being processed with all other challenges, received either from the source or from other “friends”. Challenges that lose the competition are cleared from the queue, ending up in the sink.

3.3 Global Dynamics

The purpose of the model is to better understand the self-organization of the network of agents towards increasingly efficient extraction of benefit from the situations. At the lowest level, this self-organization appears as the evolution of a differentiated network of links, where links that are more beneficial are reinforced and less beneficial ones weakened. Our working hypothesis is that this evolution will increase *coordination* between the agents (Heylighen, 2013). The basic components of coordination we expect to see are: *division of labor* (distribution of challenges to the agents most skilled to deal with them), *workflow* (efficient sequencing of processes), and *alignment* of needs (agents linked with those having similar or complementary preferences).

To test whether local reinforcement of links will lead to globally more efficient activity, we need a measure of such efficiency. The measure is simply the total benefit extracted by all the agents in the network for a given (sufficiently diverse) amount of challenges generated and injected into the network. The benefit extracted by the network of agents can be compared with the benefit extracted by the same agents when they are not connected in a network, i.e. when they only receive challenges from the generator, and not from any linked agents. The surplus in benefit created by the network represents the distributed intelligence of the system.

The learning algorithm makes it possible for the network to develop from scratch, by assuming that the initial state corresponds to a network where every agent is connected to every other agent with a very small link weight. Challenges can propagate randomly along this totally unstructured network, albeit initially providing only a tiny surplus. However, links that happen to provide benefit to the receivers will quickly be reinforced, and thus a structure should emerge and develop.

4. Formal specification of the model components

After having sketched an intuitive summary of the architecture and expected behavior of the model system, we will now explain and define the components and algorithms with the necessary detail so that they can be programmed into a simulation.

4.1 Challenges

4.1.1 Situations and agents

The basic components of the model are situations $s \in S$, and agents $a \in A$

Situations are passive, undergoing processing by the agents, who are active. During processing a situation's state is changed (while the agent remain invariant). Both situations and agents can be represented as nodes in a network (Veitas, 2012). *An agent is able to process a situation if and only if there exists a link between the situation node and the agent node.* This link represents an *encounter,* conjunction or "joint presence" of situation and agent.

What changes during propagation is the conjunction between a situation and the agents that it "visits". A situation s "moving" from agent a_1 to agent a_2 could be said to cut its initial link to a_1 and replace it with a new link to a_2. This shift can also be interpreted as agents moving from situation to situation. Both interpretations—situations moving between agents, and agents moving between situations—are in a sense equivalent, the one being a dual of the other (Heylighen, 2012b).

The first interpretation is more intuitive if we think about emails being forwarded from person to person, the second when we think about a person surfing the web from page to page. In both cases, what actually changes is the link between a situation (email or web document) and a person encountering that situation (reading the document). This more general interpretation allows us to deal with ambiguous cases, like a Facebook post appearing simultaneously on the accounts of several people in the same social network: has person A sent the post to persons B, C, D…, or has person B discovered the post that person A left on the web for his friends?

When a situation is "distributed" across different agents a_2, a_3, a_4, … or "divided" into different copies, this means that it has established multiple links, one with each agent. Indeed, a situation can be linked to more than one agent, e.g. because all agents have access to the same webpage, or because multiple copies of the same email have been sent. The number of links for a given situation allows us to monitor how widely situations spread across the network of agents, and to determine their eventual distribution (number of situations vs. number of agents having encountered that situation). The most "successful" situations may potentially reach all agents, thus producing the equivalent of a "global awareness" or "collective consciousness" for the system. For example, a situation representing climate change is likely to eventually become a challenge for all people on this planet, although this challenge is likely to be more intense for some than for others.

4.1.2 Vector representation of states

The states of a situation s are represented as vectors, i.e. lists of real numbers: $\mathbf{s} = (s_1, s_2, \dots s_N)$, $s_\mathrm{i} \in \mathbb{R}$. In principle, therefore, the state space of all possible situations forms a Euclidean, N-dimensional space: $S = \{\mathbf{s} \mid \mathbf{s} \in \mathbb{R}^N\}$. This has the advantage that classic techniques from linear algebra and geometry can be used to represent situations and their transformations without any apparent loss of generality.

It also means that situations can vary in an infinitely extended state space with a large, and extensible number of dimensions. This should allow us to capture the many subtleties and complexities of real-world situations. On the other hand, to keep the computational load in check, we will require that the vectors remain sparse, so that few of the dimensions are actually needed for the calculations (see next 4.1.3). Working with sparse vectors allows a more qualitative understanding of the model, as the few non-zero components of a vector represent specific features that the agent needs to deal with.

A more general representation would also allow unknown or unspecified quantities in the list of components, e.g. $s_i = \#$ or $s_i = ?$, as used in classifier systems (Holland, 1992). However, this would require a special algebra specifying the rules for adding or multiplying such unknowns.

4.1.3 Valence interpretation of vector components

The different components of the vector represent the dimensions, degrees of freedom or aspects of an agent's situation that have potential *valence* for that agent. This means that their value somehow affects the agent's well-being or fitness—positively or negatively. For example, temperature, calorie content, content in vitamin E, beauty, knowledge, …, could be dimensions of the challenge space.

By convention, the 0 point of each degree of freedom is assumed to be the value that is neutral (i.e. minimally satisfactory) for the *average* agent. Thus, the nil vector: $\mathbf{0} = (0, 0, \dots, 0)$ represents a neutral situation that constitutes no challenge: the situation is acceptable for the agent in all respects, and therefore the agent is not motivated to act on it.

Positive numbers represent opportunities or resources from which a benefit (increase of fitness) may be extracted. Negative numbers represent problems or disturbances that may inflict a penalty (reduction of fitness) on the agent if they are not adequately dealt with. We assume that phenomena by default tend to be neutral: neither beneficial nor problematic. Therefore, typical situation vectors are *sparse*: most of their components are zero.

4.1.4 Rival and non-rival components

The components of a situation vector can be divided into two categories: *rival* and *non-rival*. These represent respectively *material* and *informational* aspects or resources. The difference is that rival components are necessarily reduced in absolute value when the agent extracts benefit from them (because the material resource is "consumed" by the agent), while the non-rival ones maintain their value even after they have delivered benefit (because information remains available even after it has been used). For simplicity, the first M ($< N$) components will be used to represent rival quantities, while the last N - M components represent the non-rival ones.

4.1.5 Needs and Challenges

Agents a_i differ in their needs and desires. These personal requirements can be formalized by an agent-dependent need vector $\mathbf{n}(a_i) \in S$. This represents the situation that would be fully satisfactory for the agent a_i, as contrasted with the average agent for whom the need vector by definition is **0**. The challenge **c** for that agent is then defined as the difference between its need **n** and the actual situation **s**:

$$\mathbf{c}(a_i) = \mathbf{s} - \mathbf{n}(a_i)$$

The agent is motivated to transform the actual situation into the ideal situation that completely fulfils its need: $\mathbf{s} \rightarrow \mathbf{s}^* = \mathbf{n}(a_i)$. This is equivalent to reducing the challenge to zero: $\mathbf{c}^*(a_i) = 0$

4.1.6 Intensity

The intensity of a challenge $||\mathbf{c}||$ is measured as the size of the vector. It represents the amount of benefit that can potentially be extracted from the challenge, and therefore the importance of the challenge. This can be calculated as a sum of the absolute values of its components:

$$||\mathbf{c}|| = \sum_{i=1}^{N} |c_i|$$

4.1.7 A priori penalty

The intensity $||\mathbf{c}||$ of a challenge **c** is independent of the sign (+ or –) of its components c_i. However, the sign has an important meaning in the model: it distinguishes problems or deficiencies (the agent receives *less* than it needs: *negative challenge*) from opportunities or resources (the agent receives *more* than it needs: *positive challenge*). The difference is that a problem must be tackled in order to avoid a loss, while an opportunity can be exploited in order to make a gain. Successfully tackling a problem or exploiting an opportunity (i.e. processing a challenge) both provide the agent with a (positive) benefit relative to the case where the agent ignores the challenge (see further 4.2.9).

The asymmetry between problem and opportunity can be expressed in the model by automatically inflicting a loss or penalty for negative components, *before* the agent has addressed the challenge, while allowing the agent to neutralize that penalty by afterwards processing the challenge and thus extracting a benefit that compensates for the loss. That *a priori* penalty is simply the sum of the negative components:

$$Pen(\mathbf{c}) = -\sum_{\{c_i | c_i < 0\}} c_i$$

Note that the agent can choose to ignore such a negative challenge because it expects to get a higher benefit by processing a different challenge. For example, suppose it starts to rain (negative challenge). The agent can either stay under a shelter, thus neutralizing the penalty but without getting any additional benefit, or continue to walk in the rain and thus experiencing the penalty, in order to be home in time for watching a movie (and thus collecting a benefit deemed higher than the benefit lost by getting wet).

4.1.8 Generation of vectors

Need vectors are generated once for each agent. Situation vectors are generated continuously, after which each new situation is deposited in the challenge queue of one or more randomly chosen agents, independently of how many situations the queue already contains. In both cases, the vector is generated by specifying values for each of its components c_i, in such a way that most components are 0, or at least close to 0, but their distribution obeys a power law (i.e. some small values, rare medium ones, exceptionally a large one). This can be done using the following algorithm:

```
repeat for $j = 1$ to the number of agents
    repeat for $i = 1$ to $N$
        choose $x$ randomly in the interval $]0, r]$        (0 not included)
        set $c_i := \pm \beta x^{-\alpha}$
        if $c_i < \tau$, then set $c_i := 0$
    end
    deposit $(c_1, \ldots, c_N)$ in the queue/need vector of agent $a_j$
end
```

The constants r (interval size), α (power law exponent), β (constant factor) and τ (threshold) can be freely chosen, constituting parameters of the model. In general, α and β should be different for need vectors and for situation vectors, as they represent the variability between, in the one case, agents, in the other case, situations.

It is suggested that N would be of the order of 100, while the other parameters should be chosen such that the number of non-zero components ($c_i \geq \tau$) would be around 4 or 5. This results in sparse vectors, representing the fact that most challenges only affect a few valence dimensions.

4.1.9 The situation sink

There is no explicit deletion mechanism in the model, but it is assumed that if a new situation enters a queue that is already at full capacity, then the situation corresponding to the least intense challenge disappears from the queue. This means that the link between the agent and the situation is cut. Situations that no longer are linked to agents are added to the *sink*, which is simply a registry that notes the challenge, the time at which it was cleared, and the agent from whose queue it was cleared. This trace of clearing activity can later be examined for interesting patterns (e.g. situations deleted later may be more intense on average than those eliminated in the earlier stages, because more intense challenges are competing for the agent's attention).

Situations in the sink can potentially become active again by establishing a new link with a random agent. However, only situations that still offer enough benefit are likely to remain linked, as the agent would clear any situation with a too low intensity from its queue. Potentially, situations in the sink may be "regenerated" by the generator adding random valence to their components, thus increasing their probability of being processed by agents again.

4.2 Individual Agents

Since agents typically represent intelligent individuals, they are the most complex components of the model. They are likely to become more complex in later stages, as we try to simulate an increasingly sophisticated intelligence, e.g. by giving the agents the ability to learn and become better at extracting benefit. The total number A of agents in our initial simulation would be of the order of 10,000 with each agent having about 100 links to other agents. The number of agents can be multiplied in later simulations (up to billions), depending on computational constraints. However, the number of links with a non-negligible weight should remain of the same order, at least for human-type agents who can be assumed to obey the anthropological observation according to which the number of people an individual can really know is about 150 (Hill & Dunbar, 2003).

4.2.1 Challenge queue

A *challenge queue* $Q(a) = (\mathbf{c}_1, \dots \mathbf{c}_q)$ is a finite list of challenge vectors received by an agent a, either from the generator or from other agents. Its capacity is limited to the number q (which may depend on the specific agent, so as to represent differences in memory or attention span between individuals).

In the queue, challenges are normally ordered by intensity: the most intense at the top of the list, the least intense at the bottom. This makes it easy to determine whether an incoming challenge should be added to the queue: if its intensity is smaller than the one at the bottom of the queue, it can be ignored; otherwise, the bottom one should be cleared, and the new one should be added at the right position in the queue.

4.2.2 Skill vectors

We assume that agents not only have different needs, but different abilities to satisfy their needs. These abilities depend on the way they process challenges, and thus extract benefit. An agent does not know precisely how well it is able to process a challenge until after it has done so. However, we assume that an agent a can estimate the benefit it may be able to extract on the basis of its previous experience. This experience is accumulated in the *skill vector* $\mathbf{k}$, which is a weighted average of all challenges $\mathbf{c}_i$ the agent has processed, weighted by the benefit $B(\mathbf{c}_i)$ extracted:

$$\mathbf{k}(a) = \frac{1}{K}\sum_{j=1}^{K} B(a, \mathbf{c}_j)\mathbf{c}_j$$

This means that the better the agent was at extracting benefit from a challenge, the more this challenge will contribute to its skill vector.

4.2.3 Selection from queue

Challenges in the queue are selected for processing on the basis of their *expected benefit*. This is a function of their intensity $||\mathbf{c}||$ (representing the maximum benefit extractable), the degree to which the agent trusts that the challenge is real, and the degree to which the agent feels capable of extracting this benefit. This capability estimate $Cap(a, \mathbf{c})$ can be calculated on the basis of the cosine similarity between the challenge $\mathbf{c}$ and the skill vector $\mathbf{k}$: the more a challenge $\mathbf{c}$ resembles challenges that the agent a has dealt with successfully in the past, the more success the agent can expect from processing this new challenge.

$$Cap(a,\mathbf{c}) = \frac{\sum_{i=1}^{N} k_i(a) \cdot c_i}{\sqrt{\sum_{i=1}^{N} k_i(a)^2} \cdot \sqrt{\sum_{i=1}^{N} c_i^2}}$$

The trust $Trust(a, \mathbf{c}) \in [0, 1]$ of the agent in the source of the challenge is a monotonically increasing function of the weight of the link along which the challenge came in. If the source is the generator, the trust is maximal: $Trust = 1$.

The overall formula for expected benefit is then:

$$B_{exp}(a,\mathbf{c}) = ||\mathbf{c}||.Cap(a,\mathbf{c}).Trust(a,\mathbf{c})$$

After calculating this value for all challenges in its queue, the agent will select the challenge with the highest value for processing.

4.2.4 Processing

Given that the challenges are vectors, a simple and general way to process them is by means of a linear transformation that maps the initial challenge **c** onto a processed challenge $\mathbf{c}^*$. This can be represented as a multiplication of the corresponding vector with a processing matrix **M**(*a*):

$$\mathbf{m}: C \rightarrow C: \mathbf{c} \rightarrow \mathbf{c}^* = \mathbf{M}.\mathbf{c} \quad \text{with } c_i^* = \sum_j m_{ij} c_j$$

4.2.5 Correspondence with chemical organization theory

When the vectors and the matrix are sparse, most of the c_i and c^*_i will be zero. In that case, processing can also be represented via the simpler production rules (Heylighen, 2011b) or (chemical) reactions. For example, assume that the only non-zero components are c_1, c_4, c_5, c^*_2 and c^*_7. The transformation $\mathbf{c} \rightarrow \mathbf{c}^*$ can then be written as the following *condition-action rule*, *production*, or *reaction*:

$$c_1 + c_4 + c_5 \rightarrow c^*_2 + c^*_7$$

The "+" should here be read as an "and" operator, like in chemical reactions: c_1 and c_4 and c_5 together (condition) produce c^*_2 and c^*_7 (action).

In this way, we can establish a correspondence between our vector/matrix formalism and the reaction network formalism of *chemical organization theory (COT)*, which has been used to model the flow of resources and the propagation of decisions in social systems (Dittrich & Winter, 2005, 2008). The advantage of COT is that it allows a simple qualitative, algebraic analysis of the emerging closed networks of reactions ("organizations").

4.2.6 Constraints on the processing matrix

The processing should in general *reduce* the intensity of a challenge: || **M.c** || < || **c** || for most **c**. This means that the processed situation is in general closer to the agent's need vector than the initial situation—although occasional deviations may happen. This seems like a realistic depiction of an agent able to deal successfully with its environment, albeit with some errors from time to time.

This can be implemented using a matrix whose eigenvalues $\{\lambda_i(a) > 0 \mid i = 1, \ldots, N\}$ are generally smaller than 1, and rarely much bigger than 1. The condition can be formulated by requiring that the average of the eigenvalues be smaller than 1:

$$0 < \bar{\lambda}(a) = \frac{1}{N}\sum_{i=1}^{N} \lambda_i(a) < 1$$

This means that the trace of the matrix **M** (which equals the sum of the eigenvalues) should be smaller than *N*: $0 < \mathrm{tr}(\mathbf{M}(a)) < N$.

The positivity of the eigenvalues assures that the challenges are "relaxed", i.e. brought closer to zero either from above (positive valences) or from below (negative valences), but not allowed to change sign (e.g. from positive to negative) as they might with a negative eigenvalue.

4.2.7 Generation of processing matrices

Every matrix **M**(*a*) can be decomposed into its eigenvalues and eigenvectors:

$$\mathbf{M} = \mathbf{E}\Lambda\mathbf{E}^{-1}$$

Here **E** is the $N \times N$ matrix whose i-th column is the basis eigenvector $\mathbf{e}_i$ of M and $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues $\lambda_i(a) > 0$. To produce $\boldsymbol{\Lambda}$, we generate a list of N randomly chosen eigenvalues, obeying the above condition 4.2.6.

We then generate an eigenvector $\mathbf{e}_1 = (e_{1i})$, by randomly choosing values for its components e_{1i}, making sure it is sparse (i.e. most $e_{1i} = 0$) and normalized (i.e. its length = 1). We then generate a next eigenvector $\mathbf{e}_2$ in the same way, however, with the additional condition that it is orthogonal to the previous eigenvector(s). This is repeated until all eigenvectors and thus the matrix **E** is specified. The decomposition formula is then used to produce the matrix **M** that will characterize the processing by the agent.

4.2.8 Agent learning

In a later stage, we would like the agents to acquire the capability to learn, i.e. to become better in processing challenges on the basis of their previous experience. A simple way to achieve this is by noting that the matrix representation for processing is equivalent to the simplest type of neural network, i.e. a feedforward network where the input layer is directly connected to the output layer. Such a network can learn by adjusting the strength of its connections based on the degree to which its actual output deviates from its ideal output.

The matrix component m_{ij} represents the strength of the link between the node i in the network's input layer, and the node j in the network's output layer. The challenge vector $\mathbf{c} = (c_i)$ corresponds to the initial activation of the input layer, the processed vector $\mathbf{c}^*$ corresponds to the resulting activation of the output layer. The nil vector **0** corresponds to the ideally expected activation of the output layer. This allows us to apply the delta learning rule to calculate the change in value Δm_{ij} for the matrix components:

$$\Delta m_{ij} := m_{ij}(t+1) - m_{ij}(t) = \gamma c_i (0_j - c^*_j) = -\gamma c_i c^*_j$$

γ is here the learning constant, which determines the rate of adaptation of the matrix to experience: $0 < \gamma < 1$, and typically $\gamma << 1$. What the delta (difference) rule does is to reduce the values of the matrix elements with an amount proportional to the error made in the processing, i.e. the degree to which the processed challenge is different from nil (which is the vector representing the equality between situation and need). After several applications of the delta rule to adjust the matrix components, the average "error" made by processing should diminish, which means that the reduction of the challenge should become stronger, and the extracted benefit higher.

4.2.9 Calculation of benefit

The benefit extracted by an agent a after processing a challenge **c** is defined by the following formula:

$$B_{proc}(a,\mathbf{c}) = \sum_{i=1}^{N} (|c_i| - |c_i^*|)$$

Note that while the agent aims for a positive benefit, the benefit can be negative or zero. This can happen when the components of $\mathbf{c}^*$ are not closer to 0 than the components of **c** (the agent has failed to exploit a positive deviation, or to reduce a negative deviation from the neutral state). This could happen for example if **c** is an eigenvector with eigenvalue $\lambda \geq 1$.

The total benefit the agent *a* gets at time *t* after tackling a challenge **c** is the processing benefit minus the *a priori* penalty (see 4.1.7): (Note that the penalty may have been inflicted by a different challenge $\mathbf{c}^{\sim}$ that arrived at time *t*, but was not processed)

$$B(a,t) = B_{proc}(a,\mathbf{c}(t)) - Pen(a,t)$$

We need to add the condition that an agent cannot extract benefit from the non-rival components of a situation that it already encountered at an earlier stage (as confirmed by the presence of the situation identifier in the agent's memory). Otherwise, because a non-rival component is never reduced, agents could extract an unlimited amount of benefit from a repeatedly circulating challenge. This would be equivalent to a person each time again profiting from the same piece of information.

4.2.10 Agent Fitness

Each time an agent extracts benefit from a challenge, this benefit is added to its previously collected benefits. Their sum total is the agent fitness *F*(*a*):

$$F(a,T) = \sum_{t=1}^{T} B(a,t)$$

Fitness will in general increase throughout the agent's lifetime, but can temporarily decrease when the benefit is negative.

In the present model, fitness does not have any effect on the dynamics of the system, but later versions may use it by giving the fitter agents a higher capability to act (e.g. by allowing them to process more challenges per time unit). Fitness can also be used to eliminate the least fit agents, thus simulating adaptation by natural selection.

Fitness is an interesting measure for further analysis of the way it is distributed across the agents. There is no reason to assume that different agents would be equally fit, given that they have different needs, different skills, and difference positions within the network. However, we might hope that fitness would not be distributed too unevenly, because once fitness becomes a resource for processing, we should expect a *Matthew effect*, i.e. a positive feedback development in which the "rich" (fit) only become richer, while the poor become poorer (until they die…).

4.2.11 Agent IQ

Next to the fitness of an agent *a*, which is a measure of its actual performance in extracting benefit, we may consider the general intelligence or IQ of an agent, as a measure of its potential ability in extracting benefit. The first will depend on the actual challenges received, which in turn depend on the agent's "social neighborhood". The second should be independent of the agent's environment. A measure can be based on the challenge reducing power of the processing matrix **M**. This power increases as the average of its eigenvalues (see 4.2.6), and thus its trace, decreases. This leads us to the following measure which would produce an IQ of 100 for an average eigenvalue $\bar{\lambda} = 0.5$, while varying between the limits of 0 and 200 for $1 > \bar{\lambda} > 0$.

$$IQ(a) \;=\; 100\left(1 - \frac{\mathrm{tr}(\mathbf{M}(a))}{N}\right) + 50$$

While generating the processing matrices, we can impose the additional constraint, characteristic of true IQ measures, that for an average of 100, the standard deviation on IQ should be 15.

With such a measure, it becomes possible to determine the relation between individual agent intelligence and the distributed intelligence *D* of the network (see 4.4.3). At first sight, *D* should increase if we would increase average *IQ*. Yet, the effect may be less strong than one would expect, at least if the simulation would mirror empirical observations of a collective intelligence quotient (Woolley, Chabris, Pentland, Hashmi, & Malone, 2010).

The measure can also be used to establish in how far individual fitness *F*(*a*) correlates with individual intelligence *IQ*(*a*). A poor correlation would indicate a stronger effect of environmental factors (e.g. having helpful friends, an effective social network, or access to powerful resources) than of intrinsic abilities. It is interesting to examine whether the development of distributed intelligence would increase or decrease this correlation.

4.2.12 Agent mood

The agent model can be extended with a simple representation of emotion or mood, expressing how positive or negative the agent feels after processing a series of challenges. The principle is that *Mood*(*a*, *t*) improves when the agent *a* acquires a positive benefit *B*(*a*,*t*), and in particular if that benefit is larger than the expected benefit $B_{exp}(a,t)$. Mood worsens when the agent is penalized, or makes less benefit then expected. Moreover, recent experiences of benefit have a higher weight than older experiences. This can be expressed by the following formula:

$$Mood(a,t) = \sum_{j=1}^{T} c^j (B(a,t-j) - B_{exp}(a,t-j))$$

The parameter $0 < c \leq 1$ represents here the degree of exponential decay of the importance of older experiences ($t - 1$) with respect to newer experiences (t). T represents the total number of experiences over which the mood is calculated. This can vary between unlimited and relatively small, because the exponential decay ensures that very old experiences ($T >> 1$) anyway contribute very little.

4.2.13 Function of mood

According to the broaden-and-build theory (Fredrickson, 2004), positive emotions broaden people's domain of interest, motivating them to explore and thus discover new resources on which they can build further. Negative emotions, on the other hand, focus the attention on the immediate problems, narrowing the field of consciousness, and thus reducing the risks of neglecting dangerous situations. In the present model, this could be formalized by making agents in a positive mood more willing to take on risky or novel challenges, i.e. challenges for which their capability and trust estimates are low. This could be expressed by replacing the formula for expected benefit B_{exp} (4.2.3), which is used as the criterion for challenge selection, by the following, mood-dependent function:

$$B_{exp}(a,\mathbf{c}) = \|\mathbf{c}\| \cdot \left(\frac{\text{Max}(|Mood(a)|) + Mood(a)}{2\text{Max}(|Mood(a)|)} + Cap(a,\mathbf{c}) \cdot Trust(a,\mathbf{c}) \right)$$

Max(|*Mood*(*a*)|) here represents the maximum of the absolute value of the mood as experienced by agent *a*. This means that the first term (quotient) in the sum becomes 1 when the mood is maximally positive, and 0 when it is maximally negative. The result is that in a fully negative mood, the formula reduces to the original, highly selective formula, while in a fully positive mood the absolute intensity of the challenge plays a much larger role than the agent's trust in its own abilities and the reliability of the sender.

4.2.14 Agent reputation

The reputation of an agent a_i (i.e. the degree to which it is influential in the network) can be defined as the total degree of trust that other agents have in a_i:

$$Rep(a_i) = \sum_j w_{ij}$$

An agent's reputation could be used as a preliminary measure of trust in case an agent receives a challenge from an agent it has never interacted with before.

4.2.15 Agent social capital

An agent's "social capital" can be defined as the complement to its reputation, i.e. as the degree to which other agents are willing to help it by sending it (potentially beneficial) challenges:

$$Soc(a_i) = \sum_j w_{ji}$$

An agent with a high social capital has a strong social network of friends to rely on, and as such is likely to accumulate more fitness than an agent with low social capital. Note that social capital measures incoming link weights, while reputation measures outgoing link weights. The two measures are correlated only to the degree that links are symmetrical, which is not in general the case.

4.3 Propagation and Network

4.3.1 Preparing the challenge for propagation

After processing, the challenge left for propagation **c'** is reconstituted from the processed version **c*** for the rival components and the original version **c** for the non-rival components:

$$\mathbf{c'} = (c_1^*, \ldots, c_M^*, c_{M+1}, \ldots, c_N)$$

This outgoing challenge is transformed back into a situation **s**', by adding the agent's need vector that was subtracted when determining the original challenge:

$$\mathbf{s}' = \mathbf{c}' + \mathbf{n}(a)$$

If this situation is sent to $k > 1$ agents, the rival components need to be divided by k, since rival resources are by definition conserved, meaning that their sum total must remain the same (when it is not being processed). This produces the following outgoing situation:

$$\mathbf{s}'' = \left(\frac{s'_1}{k}, \ldots, \frac{s'_M}{k}, s'_{M+1}, \ldots, s'_N\right)$$

The benefit left for the receiving agent will be calculated on this new vector **s**''. In general, this benefit will be smaller than the original benefit because of the reduction of the rival components. The outgoing situation **s**' will be added to the queues of a selected number of agents linked to the agent *a*, where it will be evaluated as to its expected benefit for each receiving agent. Given the different link weights, need vectors and skill vectors, this expected benefit could be very different for the different agents. This implies that only some of the receiving agents are likely to further process this challenge.

4.3.2 Preference vectors

Any challenge in a queue can be transmitted to another agent, or not. Such propagation can happen with or without processing. The decision about which challenge to transmit to which agent should ideally depend on the expected benefit of the challenge for the receiving agent. This depends on the potential benefit left after processing, i.e. $||\mathbf{s}'||$, and potentially on the need and skill vectors of the receiving agent, which however are not accessible to the sending agent. To make this realistic, we need to assume some (imperfect) mechanism by which the sending agent may get to "know" the desires and abilities of its contacts.

A promising approach is similar to the one we used for computing skill vectors (4.2.2): let an agent aggregate preference vectors $\mathbf{p}(a_i)$ which represent the kind of challenges actually processed by each of its contacts a_i. The preference vector is simply the average of all vectors it sent to the contact that were effectively processed (i.e. that were not deleted from the queue), together with its own skill vector **k**:

$$\mathbf{p}(a) = \frac{1}{K+1}\left(\sum_{j=1}^{K} p_j(a)\mathbf{c}_j + \mathbf{k}\right)$$

Here $p_j(a) = 1$ if agent a processed challenge $\mathbf{c}_j$, and $p_j(a) = 0$ otherwise. The inclusion of **k** in the average is useful for the situation where the receiving agent has not processed any received challenge yet. It represents the assumption that the receiver's skills are similar to the sender's by default.

4.3.3 Decision to propagate

The sending agent calculates the cosine similarity (see 4.2.2) $Sim(\mathbf{s}', \mathbf{p}(a_i))$ between the outgoing situation vector and the vector $\mathbf{p}(a_i)$ representing the assumed preferences for each of its contacts, and multiplies it with the situation intensity to calculate the expected benefit for each potential recipient:

$$B_{rec}(a_i, \mathbf{s}') = ||\,\mathbf{s}'||.\ Sim(\mathbf{s}', \mathbf{p}(a_i))$$

It sends the situation to all agents for which $B_{rec} \geq b$, where the propagation threshold b is a (potentially agent-dependent) parameter of the model. Contacts whose B_{rec} is smaller than the threshold do not get a copy of the situation. If there are no potential recipients above the threshold, the situation is cleared to the sink.

4.3.4 Cost of actions

In a later stage of the modeling, we may attribute different costs to the different actions performed by the agents, such as evaluating challenges in its queue, or propagating challenges. These costs can be expressed as penalties (negative benefits) that reduce the benefit extracted by processing the challenge. This is likely to lead to non-trivial trade-offs, in which the agent e.g. may try to increase its fitness by evaluating more challenges for their expected benefit contribution, but at the same time lose benefit because the cost of evaluation reduced the benefit collected. Thus, costs will affect the agent fitness, individual and link learning, and the overall distributed intelligence, making the model economically and psychologically more realistic by taking into account the limited cognitive or material resources of real people.

4.3.5 Forum agents

For a more realistic representation of Internet technologies, we wish to introduce a special type of agent that merely stores and propagates challenges, without any selection or processing. Such a forum agent represents a local communication medium, such as a webpage, a wiki, a mailing list, or a discussion forum, that functions as an intermediary or conduit in which challenges can be "written down" for other agents to "read" (stigmergic communication, (Heylighen, 2012b)). A forum agent is passive in the sense that it does not actively intervene in the challenge or extract benefit from it. It merely makes it available to other agents to which it is linked.

The formal structure of a forum agent is similar to a normal agent, except that its need vector is nil: $\mathbf{n}(a) = \mathbf{0}$, and its processing matrix is the identity matrix: $\mathbf{M}(a) = \mathbf{I}$. Its queue typically has a larger capacity q than the one of ordinary agents, and its number of links is unlimited.

Forum agents can be linked to other agents (active or passive), and these links will undergo the standard reinforcement if the challenges propagated along them turn out to be beneficial for the receiving agents.

4.3.6 Payment and Currency

In an economic interpretation of the network, agents should be able to pay for the beneficial challenges that they get from others. The currency is simply raw benefit B that is transferred from the buying agent a_i to the selling agent a_j. This means that a_i's fitness store $F(a_i)$ (which here plays the role of accumulated benefit, or "capital") is reduced with the amount P of the payment, while a_j's capital $F(a_j)$ is increased with the same amount. For a realistic model of a buying transaction, in return for the payment a_i passes on a challenge $\mathbf{c}$ from which a_j expects to extract a benefit larger than the cost of payment:

$$B_{exp}(a_j, \mathbf{c}) > P$$

If this expectation is realized after processing ($B(a_j, \mathbf{c}) > P$), then both agents' fitness will have increased through the transaction. Thus, agents are "motivated" to engage in this kind of economic transactions in which a valuable challenge is sold in return for raw benefit. However, note that expectations can be wrong, and that the buying agent may make a loss because it overestimated its capability to extract benefit from the purchased challenge.

4.3.7 Markets

Market mechanisms can be modeled by assuming that different agents a_i can "bid" for a challenge $\mathbf{c}$, i.e. propose different amounts of benefit depending on how much benefit $B_{exp}(a_i,\mathbf{c})$ they themselves hope to extract from the challenge. The challenge would then be sold to the highest bidder, which would normally be the agent with the highest need for the challenge and/or the highest skill in extracting benefit from it.

To model a full market, selling agents should be able to offer their challenges not only to their contacts, but to a large, anonymous group of agents willing to bid for it. This could be achieved by "publicizing" the sale via a forum agent. The highest bid proposed by the agents linked into the forum (here playing the role of a "marketplace") would then become the *price* of the challenge.

4.3.8 Sensor agents

A sensor agent is the equivalent of a measuring or observation apparatus, such as a thermometer or automatic camera, which takes information from outside the network and injects it into the network. Like a forum agent it is passive in that it does not select challenges, but merely propagates them. Unlike a forum agent, it receives its challenges straight from the generator, not from other agents. It may process these challenges (e.g. filter out irrelevant components), but without extracting any benefits, as it has no needs. Its function is to redistribute these challenges to a selected number of output agents, thus making external challenges more widely available.

4.3.9 Tool agents

A tool agent is the equivalent of a program, device or instrument that can process situations when used by an active agent, the "user". The user simply passes on the situation to the tool agent who processes it and returns the processed situation to the user. The benefit extracted by the tool agent also goes to the user agent. Tool agents have no need vectors, only processing matrices. Like forum and sensor agents, they do not select challenges, but accept everything they receive from users and process it in the order of receipt. User agents strengthen their links with tool agents in proportion to the amount of benefit they receive through them, just like they adjust their links with other agents (4.3.12). This is the equivalent of humans learning which tools or technologies are most effective.

In a later stage, tool agents should be able to evolve via variation (small random changes in the matrix), recombination (linear combination of different matrices) and selection (agents with a higher fitness are allowed to reproduce, agents with a low fitness are eventually eliminated). Fitness in this case could be measured either as the sum of benefit produced (4.2.10), or as the social capital of the agent (4.2.15). This, coupled with the rewiring of links between user agents and tool agents, provides a model for technological innovation within an ecosystem of users and tools. Its likely long-term effect is the development of a technology that is *omnipresent* (all users linked to powerful tools), *omnipotent*/*omniscient* (tools available that can effectively process any kind of situation, respectively for rival and for non-rival components), and *omnibenevolent* (tools available for satisfying any need) (see Heylighen, 2014).

4.3.10 Links

The agents $A = \{ a_i \}$ form the nodes in a directed, weighted network, with links $l_{ij} = (a_i, a_j)$, and weights: $w(l_{ij}) = w_{ij} \in [0, 1]$. There is no a priori constraint on which node can be linked to which other node. Networks can have various topologies, characterized by properties such as clustering, small-worlds and power-law degree distributions (Albert & Barabási, 2002). These are normally not imposed, but the result of self-organization.

4.3.11 Interpretation of weights

The weights represent the strength of the "social connections" between agents, in the sense that an agent will pay more attention to a challenge received through a strong link than through a weak link. This strength can be interpreted as the degree of *trust* in the seriousness of the sending agent (Van Overwalle & Heylighen, 2006), or as the willingness to follow the suggestion or accept the proposal of that agent. Concretely, it means that a challenge received through a strong link is more likely to be processed than one received through a weak link, as expressed by the formula for calculating the expected benefit of a received challenge (see 4.2.3).

4.3.12 Reinforcement of agent-agent links

The weight w_{ij} of a link l_{ij} will increase each time a challenge is transmitted successfully from sender a_i to receiver a_j along this link. Success is measured by the benefit extracted by the receiver, because benefit for a_j means that next time a_j will be more likely to accept a challenge from a_i.

$$\Delta w_{ij}(t) := w_{ij}(t+1) - w_{ij}(t) = \varepsilon B(a_j, t)$$

However, it seems better to make the reinforcement to some degree reciprocal, so that the sender a_i too would somehow benefit from a successful extraction by the receiver a_j. This can be done via an application of the "symmetry rule" (Bollen & Heylighen, 1998): a part κ (e.g. one tenth) of the reinforcement for l_{ij} is also given to the reciprocal link l_{ji}. This means that next time a_i is more likely to receive a challenge from a_j, so that a_i indirectly may benefit from sending a beneficial challenge to a_j.

$$\Delta w_{ji} = \kappa \, \Delta w_{ij}, \; 0 < \kappa < 1$$

Another possibility is to let reinforcement depend on synergy or cooperation, i.e. extraction of benefit by both sending and receiving agents:

$$\Delta w_{ij} = \varepsilon B(a_i, t).B(a_j, t)$$

Here, agents are "motivated" to establish connections with those with whom they can work synergetically. However, in this case, there is no reinforcement if one agent merely passed on the challenge without extracting any benefit.

It may be useful to also apply some version of the "transitivity rule" (Bollen & Heylighen, 1998) which reinforces indirect connections. For example, if a_1 successfully sends a challenge to a_2, who passes it on successfully to a_3, then it seems worth reinforcing the indirect link l_{13}, and not just the direct links l_{12} and l_{23}. This would help agents to extend their social network by establishing contacts with the friends of their friends. This would indirectly support preferential attachment, as agents with many incoming links will also have many friends of friends, and therefore many candidates to establish additional links with. When combined with a small number of random encounters as determined by R, such transitive coupling with preferential attachment to popular individuals appears like a realistic model of how people get to know other people.

4.3.13 Reinforcement of situation-situation links

The situations $S = \{ s_i \}$ too can be seen as the nodes in a directed, weighted network, with links $l_{ij} = (s_i, s_j)$, and weights: $w(l_{ij}) = w_{ij} \in [0, 1]$. This network represents the spatial or virtual topology of the arrangement of situations. For example, if situations correspond to webpages, the links between them correspond to hyperlinks pointing from the one page to the other; if situations correspond to objects situated in particular locations, the links represent the physical reachability between the situations.

In some circumstances, these weights may be increased each time an agent a_k encounters the situations s_i and s_j in immediate succession by following the link l_{ij}. The amount of increase obeys a similar reinforcement rules as for links between agents:

$$\Delta w_{ij} = \varepsilon' B(a_k, s_i, t).B(a_k, s_j, t)$$

Agents adding weights to links connecting beneficial situations is the equivalent of the "pheromones" that ants leave on their path from nest to food sources. A similar reinforcement is at the basis of the global brain-like web architecture originally proposed by (Heylighen & Bollen, 1996, 2002) and of the "man-hill" model for a self-organizing network of educational materials (Gutiérrez, Valigiani, Jamont, Collet, & Delgado Kloos, 2007; Valigiani, Biojout, Jamont, Lutton, & Collet, 2005). It implements a form of stigmergy, where agents are stimulated to perform useful works by the situations and the "pheromone traces" they encounter, leaving further traces to stimulate subsequent agents.

At present such a reinforcement does not yet exist in most ICT networks, but there is good reason to believe that introducing it would significantly boost the development of distributed intelligence.

4.3.14 Generation of links

Typical network generating algorithms create links between random pairs of nodes, possibly modulated by some preference function, such as the preferential attachment to highly linked nodes that has been shown to produce a power-law distribution of node degrees (Albert & Barabási, 2002). These methods can be generalized by interpreting the network as the self-organizing medium for challenge propagation.

Assume that initially, when there are no links in the network yet, situations move from agent to agent by random jumps: each situation is linked to one agent randomly chosen in A. If this agent decides to process the corresponding challenge, a link is created from the sending agent to the receiving one, with a link weight proportional to the benefit extracted, using the reinforcement formula above with $w_{ij}(t) = 0$. Similarly, if the same agent processes two subsequent situations, a link can be created between these situations. As the number of links created in this manner increases, challenges will propagate preferentially along the links, and the probability R of random jumps decreases proportionately with the total outgoing link weight:

$$R(a_i) = \frac{1}{1 + r_0 \sum_j w(l_{ij})}$$

This means that random jumps (which potentially create new links) do not completely disappear. This allows the system to maintain a degree of flexibility or creativity by exploring new configurations. The parameter r_0 determines the frequency of random jumps: the larger it is, the more rarely jumps occur.

4.4 Global Dynamics

4.4.1 Challenge diffusion

When the generator adds a new challenge to the queue of an agent, this challenge can potentially spread across the network via repeated transmission to linked agents. This process is similar to spreading activation in neural network models: an increasing number of nodes receive a decreasing amount of activation, until the activation becomes lower than the threshold value required for further propagation. In an agent network, the "degree of activation" corresponds to the intensity of the challenge, which also tends to decrease as the challenge is reduced by subsequent agents.

However, because of their different needs, reduction by one agent may actually intensify the challenge for another agent, so the decrease is non-monotonic. There is also no fixed threshold for propagation, as the decision to propagate will depend on contextual factors, such as remaining intensity after processing and preference vectors. This means that a challenge too weak to be transmitted by one agent may still be transmitted by another agent. Moreover, there is no reduction of the non-rival components, so challenges scoring high in these components may continue to spread until they have reached every agent. Therefore, the process of challenge propagation appears more complex than the process of spreading activation, although we may expect a roughly similar short-term dynamics of limited diffusion depending on the initial intensity ("activation") in the input node.

4.4.2 Long-term dynamics

The long-term dynamics of the network depends on the generation and reinforcement of links that change the topology of the network. In a closed system, this dynamics will normally settle in an attractor—although this may take a very long time given the complexity of the system. The present model, on the other hand, assumes an open system with an unpredictable input of randomly generated challenges and connections. This means that strict attractors (in the sense of configurations the system can enter but never leave) are unlikely to be reached. On the other hand, "fuzzy attractors" (in the sense of configurations the system has high probability of entering but low probability of leaving) are likely to emerge. These approximate attractors are likely to be more robust than the strict attractors, given that they are constantly perturbed by incoming challenges, so that they undergo variation and selection for stability in the face of random disturbances.

An analysis of simulation results should allow us to determine the typical properties of these approximate attractors, such as network topology, global ability to deal with challenges, division of labor among the agents, "supply chains" between agents implementing an efficient workflow, distribution of fitness across the agents, etc.

4.4.3 Distributed intelligence measure

The most important emerging property for our purpose is the distributed intelligence of the system. This depends on the average rate $\overline{B}$ of benefit extraction:

$$\overline{B}(t) = \frac{1}{TN}\sum_{i=1}^{N}\sum_{j=0}^{T} B(a_i, \mathbf{c}(a_i, t+j))$$

Here, $\mathbf{c}(a,t)$ represents the challenge processed by agent a at time t, while T is the total number of time steps over which the average is calculated.

For a real measure of distributed intelligence, this should be compared with the average benefit extraction $\overline{B}^{np}$ when no propagation occurs. This is calculated by the same formula but in a configuration where agents only process challenges received from the generator, not from their contacts. $\overline{B}^{np}$ can be interpreted as the average individual performance of the agents in the system. The full measure of distributed intelligence D then becomes:

$$D(t) = \frac{\overline{B}(t)}{\overline{B}^{np}}$$

We here assume that $\overline{B}^{np}$ is not dependent on the time t, given that without propagation, the network does not change, and therefore its extraction abilities remain constant. Of course, since $\overline{B}$ and $\overline{B}^{np}$ are averages over a large number of random challenges, they will vary depending on the specific sequence of challenges generated. However, by choosing N and T large enough, variations across sequences should be minimized because of the law of large numbers. Simulation experiments should allow us to find a good value for these parameters.

4.4.4 Dynamics of distributed intelligence

The fundamental hypotheses of the model can be stated as:

$$D(t) \geq 1$$

$$\frac{dD(t)}{dt} > 0$$

$$\lim_{t \to \infty} \frac{dD(t)}{dt} = 0$$

This expresses the fact that, while the network adapts, its distributed intelligence *D(t)* will increase, until it settles down into an approximate attractor where further evolution is no longer significant. In other words, distributed intelligence reaches its maximum for this particular configuration of agents, challenge generation and link learning rules.

The specific rate and rate change of distributed intelligence increase should become clear from the simulation results. A plausible scenario is that the rate of increase starts small, as agents still need to try out many random connections without much gain in benefit, then accelerates as complementary clusters of synergistically working agents are formed, and then slows down again as the system settles down in the attractor. The overall shape of the function $D(t)$ may well resemble an S-curve or logistic growth process, which is typical for self-organizing processes characterized by an initial positive feedback, but a limited "carrying capacity" (the system cannot extract more resources than are actually present in the challenges). The steepest point of such a curve could then be interpreted as the critical transition t_c to the global brain regime:

$$\frac{d^2D(t_c)}{dt^2} = 0$$

Although this seems unrealistic given the present assumptions, if the transition is steep enough, it could be approximated by a singularity:

$$0 \ll \frac{dD(t_c)}{dt} \simeq \infty$$

The most important quantity coming out of the simulation will be the final distributed intelligence:

$$D_\infty = \lim_{t \to \infty} D(t) > 1$$

For a true global brain, we would expect this quantity to be much larger than 1. How large it can get, will depend on a variety of parameters and assumptions characterizing the model.

4.4.5 Optimization of parameter values

The model as we sketched it depends on a variety of free parameters (see Table 1) that are likely to influence the main outcome variable *D* and its evolution. Part of the intention of the model is to determine the parameter values for which *D* increases most. Some monotonic dependencies can already be guessed from the general dynamics of the model.

For example, *D* is likely to increase with *q*, the maximum size of the queue, because a larger queue allows an agent to select potentially better challenges for processing. However, if we would introduce a cost for examining challenges in the queue, a too large queue may decrease the agent's benefit. In this case, we can expect a trade-off between a larger number of beneficial challenges to choose from and a higher cost in deciding which one to choose resulting in an optimal value of *q* that is neither too high nor too low.

Another dependency that is likely to be monotonic in the simplest version of the model is the one on *M*, the number of rival components in the challenge vector. The smaller *M*, the larger the number $N - M$ of non-rival components, and therefore the more benefit can be extracted from these components after the challenge has already been processed by previous agents.

Other parameters are likely to have a more complex effect on distributed intelligence, which can only be established by running many simulations. One example is the learning rate ε, which should not be too low, in order not to needlessly slow down self-organization, but not too high either, in order not to forget what was learnt previously. A similar logic applies to the random jump probability r_0.

A general strategy to explore the parameter space could be the following. First, we choose simple, but still realistic starting values for all of the parameters (as proposed in Table 1), and let the simulation run until we get the final distributed intelligence D_∞. Then, we vary each parameter one by one, each time rerunning the simulation with small increments of the parameter within its range (when the range covers several orders of magnitude, increments change the value exponentially, otherwise linearly). For each parameter, we keep the value that produces the highest distributed intelligence. This optimizes all the parameters separately, but neglects the likely interactions between parameters. Since the space of all possible combinations of parameter values is much too large to search systematically, we can use the following heuristic.

We first order the parameters by the amount D gained by optimizing them individually, so that we get a list with the most "important" parameters first. We re-optimize the first one, then the second one while keeping the newly optimized value for the first one, then the third one, while keeping the newly optimized values for the first and second parameters, and so, until all parameters are re-optimized. In the next iteration, we keep the optimized values of all parameters, but again vary the first one, so as to determine its new optimized value on the background of all the others. We do the same with the second one, then the third one, and so on. In this way parameters values can to recursively "co-evolve": the one adapting to the new values of the other ones, and so on.

Parameter	Meaning	Range	Starting value
A	Number of agents	10 — 10^{10}	10 000
N	Dimensionality of challenge space	10 — 1000	100
M	Number of rival components	$0 \leq M \leq N$	$N/2$
τ	Threshold for setting components to 0	?	?
q	Capacity of challenge queue	1 — …	10
α	Power in power law distribution	?	?
β			
	Number of links per agent	10—1000	100
γ	Agent learning rate	$0 \leq \gamma << 1$	0
$\bar{\lambda}$	Average eigenvalue	$0 < \bar{\lambda} < 1$	0.5
IQ	Average agent intelligence (derived from $\bar{\lambda}$)	0 < IQ < 200	100
Forum	Number of forum agents	0 — 10^9	0
b	Threshold for propagation to potential recipient	?	?
Sens	Number of sensor agents	0 — 10^9	0
r_0	Reduces probability of random jump	0 — 1000	?
κ	Reciprocal link weight increase	0 — 1	0
ε	Link learning rate	0 — 1	0.1
T	Number of time steps for averaging	1 — …	10

Table 1: a list of parameters used in the model together with their ranges and a plausible starting value.

As a final attempt to optimize the global parameter configuration, we can explore its variations with the help of the *simulated annealing* method: we randomly vary all "optimal" parameter values at once to create a new starting point for a simulation and compute D_∞. If it is much smaller than the previously best value for D_∞ we make another big random variation of all parameters and repeat the procedure, until we get a better value for D_∞. As the value of D_∞ improves, we slowly decrease the size of the random jumps, each time keeping the combination of parameter values that gave the best result as basis for further variation, and thus try to reach a global optimum. Such a strategy will obviously require a lot of computation exploring different simulations, but it seems to be the only one with a high probability of finding a near optimal configuration…

4.4.6 Some illustrative variations

Variations of the parameter values are not only useful to find the "optimal" development of distributed intelligence, but to explore various aspects and processes characterizing a socio-technical system with distributed intelligence. The diversity of possible scenarios to investigate can be illustrated by the following elementary variations of the model.

A variant with only non-rival components ($M = 0$) would represent pure transfer of information, without any matter, energy or money being exchanged. This could be used to simulate the spread of memes ("good ideas") across the network, taking into account the self-organized (or imposed) network topologies. Given the competition for processing and propagation, only the "best" challenges are likely to spread globally throughout the network, but this may depend on a variety of local configurations and non-linear effects. This variant could also be used to model the emergence of communities of interest, since agents with similar interests (need for non-rival components) are likely to cluster together, because they derive benefit from the same kinds of informational challenges. This may help us to understand how separate subcultures emerge (Axelrod, 1997; Heylighen & Chielens, 2008; Van Overwalle & Heylighen, 2006).

A variant with only rival components ($M = N$), on the other hand, could be a model of an ecosystem in which matter and energy are propagated through the network. This could be turned into a model of a market-like economic system if the agents who receive a challenge would "pay" the sender with an amount of benefit lower than the one they expect to extract from it, but higher than the one the sender could still extract from it. The default model, on the other hand, could be interpreted as a model of a gift economy, where an agent "gives" the resources it no longer needs to its friends, but by tightening the friendship links in this way, becomes more likely to get reciprocal gifts and thus increase its welfare.

Another variation of the model would be to add a large number of sensor and/or forum agents (*Sens* >> 0, *Forum* >> 0). These represent technologies that propagate information more widely. This may allow us to simulate the effect of increasingly powerful and available ICT on distributed intelligence. We can also add more general "machine" agents (tools, technologies, robots, …) specialized in processing certain challenges. These agents have no needs, and thus basically accept any challenge that is put to them, as long as they are able to process it. The network will then self-organize to optimally make use of their specific skills to deal with specific challenges. From time to time, new types of such agents could be generated, in order to simulate the process of innovation. This is likely to stimulate a process of adaptation and reorganization of the network.

A variant in which the agents can individually learn ($\gamma > 0$) may help us to understand how social learning (getting to know interesting people) and individual learning (experiencing interesting challenges) interact. We may hope to discover a positive feedback between both effects, establishing a virtuous cycle of cognitive development eliciting more cognitive development. This would also increase average *IQ*, and indirectly distributed intelligence *D*, although the relation between *D* and *IQ* needs to be explored on its own.

5. Further exploration and development of the model

5.1 Modularity of extensions

Most parameters of the model have a specific value (typically 0 or 1) for which the complexity of the system is reduced. For example, $N = 1$ makes challenges one-dimensional, $\varepsilon = 0$ fixes the network topology, and $b = 0$ stops the agents from discriminating between potential recipients. This allows us to easily control the complexity of the model. The advantage is that if the model would behave in a way we did not expect, then we can easily pinpoint the origin of the problem by "switching off" various features ("modules") one by one via these parameter values, until the system behaves again in a way we understand. This ensures that the model can be simplified or complexified depending on the phenomena we want to model, without the modelers losing control of what is going on.

The model not only allows variations in parameter values but also variations in components, topology, and dynamical rules. This is made possible by the highly modular architecture, where the different components are defined in such a way that they can be changed without affecting the rest of the components.

For example, the matrix multiplication used for challenge processing could be replaced by a neural network, or a system of production rules, because all these processing mechanisms can work with vectors as inputs and as outputs. Similarly, the challenge generator could produce a completely different distribution of challenges and challenge components, without this affecting how the challenges would be processed and propagated by the agents. We could also decide to implement a different way to compute benefit, or a different algorithm for changing linking patterns, again without this disturbing the functioning of the other components.

Different variations can be used to explore different assumptions, subsystems, processes, or scenarios. This would allow us to investigate e.g. the role of different technologies, aspects (e.g. social, economic, or cultural), and strategic choices in the development of a global brain. Given the richness of the model, the diversity of conceivable variations is virtually unlimited.

5.2 Empirical tests

Every good scientific model should be tested sooner or later against the phenomena it purports to model—in our case the distributed processing and propagation of information on the Internet. Given the complexity of the model, the high level of abstraction or generality at which it is formulated, and the intrinsic complexity and variability of the phenomena being modelled, designing such tests is not a trivial matter. However, the modular, parameter-controlled architecture makes it possible to test different features one by one, by "switching off" all the apparently irrelevant modules while simulating a particular phenomenon, such as memetic spreading, or market self-organization.

The main issue is to find empirical data that can be reliably converted into the kind of elements used by the model, i.e. vectors, agents and links. Agents clearly correspond to human individuals, while the social links between them can be inferred with classical methods for mapping social networks, such as registering the "friend" links in Facebook. Converting messages exchanged between individuals into vectors is the least obvious requirement to fulfill. A promising way to achieve this is the method of *Latent Semantic Indexing* (LSA), which analyses the frequency of occurrence of words in documents, and uses the resulting word-document matrix to extract the most important semantic dimensions that characterize the documents (Landauer, McNamara, Dennis, & Kintsch, 2007). Mapping documents onto these dimensions produces vectors similar to the ones used in our formalism.

This makes it in principle possible to convert texts, such as emails, Facebook posts, Twitter posts, or personal profiles, into vectors. The vectors representing messages sent to individuals can then be interpreted as challenges, while the vectors representing their profile of interests can be interpreted as need vectors. Given challenges, agents, needs, and links, our model should then be able to predict significantly better than chance which challenges will be propagated to which agents. If not, we will know that some of our assumptions are erroneous and need to be corrected. If the prediction would turn out to be better than chance, but otherwise not very good, we may be able to improve it by optimizing the parameters or fine-tuning the hypotheses of the model. In this way, we can use empirical feedback to make our model increasingly more realistic and accurate.

5.3 Need for additional resources

The major practical constraint in investigating the variety of scenarios and tests that we conceive is the time necessary to gather and process data, and implement, simulate, and analyze the results for each scenario. This will obviously depend on the amount of manpower and computational power available to run these experiments: the larger the group of researchers, and the better their infrastructure, the wider the variety of cases or applications that can be investigated, and the greater the probability of finding truly significant results.

The architecture of the model is not only modular but highly integrated, as it is based on the conceptually and mathematically coherent paradigm of challenge propagation (Heylighen, 2012b). The present paper has specified the mathematical architecture, while a companion paper (Veitas, 2012) has presented a first sketch of the software architecture for the implementation. This architecture should make it easy for different people with different backgrounds to collaborate on developing different aspects of the model in a coordinated manner. Moreover, it should allow expanding the size of the simulation (number of agents, components, links, …) with several orders of magnitude, without requiring fundamental changes (the proposed Tinkerpop implementation (Veitas, 2012) in principle allows simulating a network with billions of nodes).

Thus, we hope that with a sufficiently large group of contributors having sufficient computational resources and access to real-world data the present model will grow into an increasingly realistic and concretely applicable simulation of the emerging Global Brain.

6. Conclusion

The present formalization of the challenge propagation theory of the global brain appears sufficiently complete and explicit to implement it as a full simulation. One of us (Busseniers) has already started developing a prototype implementation in Matlab, to test for bugs and unforeseen effects, while another one (Veitas, 2012) has formulated the basic architecture for a full implementation using the graph traversal language developed by Marko Rodriguez (Rodriguez, 2010, 2012). These implementations will undoubtedly teach us many lessons, initially suggesting ways to correct, refine and extend the model, eventually providing testable predictions about how the actual global brain is likely to function and evolve in different circumstances.

While the implementations are being developed and tested, the mathematical model and the underlying theory too will continue to develop on the basis of new insights. Such refinements should not interfere with the software development, as the addition of a module or a change in the functioning of a module can be easily performed while keeping the rest of the model intact. Thus, we expect the model to continue to improve at its presently fast pace, so that it should be ready for testing with real world data within the next six months or so.

7. References

Albert, R., & Barabási, A. L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, *74*(1), 47.

Anderson, P. W. (1972). More is different. *Science*, *177*(4047), 393–396.

Axelrod, R. (1997). The dissemination of culture. *Journal of conflict resolution*, *41*(2), 203.

Bernstein, A., Klein, M., & Malone, T. W. (2012). Programming the Global Brain. *Communications of the ACM*, *55*(5), 1.

Bollen, J., & Heylighen, F. (1998). A system to restructure hypertext networks into valid user models. *New Review of Hypermedia and Multimedia*, *4*(1), 189–214. doi:10.1080/13614569808914701

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, *99*(Suppl 3), 7280–7287. doi:10.1073/pnas.082080899

Corning, P. A. (1995). Synergy and self-organization in the evolution of complex systems. *Systems Research*, *12*(2), 89–122.

Dittrich, P., & Winter, L. (2005). Reaction networks as a formal mechanism to explain social phenomena. In *Proc. Fourth Int. Workshop on Agent-Based Approaches in Economics and Social Complex Systems (AESCS 2005)* (pp. 9–13). Retrieved from http://users.minet.uni-jena.de/~dittrich/p/DW2005.pdf

Dittrich, P., & Winter, L. (2008). Chemical Organizations in a Toy Model of the Political System. *Advances in Complex Systems*, *11*(04), 609. doi:10.1142/S0219525908001878

Fredrickson, B. L. (2004). The broaden-and-build theory of positive emotions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *359*(1449), 1367.

Goertzel, B. (2002). *Creating internet intelligence: Wild computing, distributed digital consciousness, and the emerging global brain*. Kluwer Academic/Plenum Publishers.

Gutiérrez, S., Valigiani, G., Jamont, Y., Collet, P., & Delgado Kloos, C. (2007). A swarm approach for automatic auditing of pedagogical planning. In *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on* (pp. 136–138). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4280973

Heylighen, F. (1991). Modelling emergence. *World Futures*, *32*(2), 151–166. doi:10.1080/02604027.1991.9972256

Heylighen, F. (1999). Advantages and limitations of formal expression. *Foundations of Science*, *4*(1), 25–56. doi:10.1023/A:1009686703349

Heylighen, F. (2001). The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, *5*(3), 253–280.

Heylighen, F. (2008). Complexity and Self-organization. *Encyclopedia of Library and Information Sciences, eds. MJ Bates & MN Maack Taylor & Francis*. Retrieved from http://pespmc1.vub.ac.be/Papers/ELIS-complexity.pdf

Heylighen, F. (2011a). Conceptions of a Global Brain: an historical review. In *Evolution: Cosmic, Biological, and Social, eds. Grinin, L. E., Carneiro, R. L., Korotayev A. V., Spier F.* (pp. 274 – 289). Uchitel Publishing. Retrieved from http://pcp.vub.ac.be/papers/GB-conceptions-Rodrigue.pdf

Heylighen, F. (2011b). Self-organization of complex, intelligent systems: an action ontology for transdisciplinary integration. *Integral Review*. Retrieved from http://pespmc1.vub.ac.be/Papers/ECCO-paradigm.pdf

Heylighen, F. (2011c). *The GBI Vision: past, present and future context of global brain research* (No. 2011-01). Retrieved from http://pespmc1.vub.ac.be/papers/GBI-Vision.pdf

Heylighen, F. (2012a). *A Tale of Challenge, Adventure and Mystery: towards an agent-based unification of narrative and scientific models of behavior* (Working papers No. 2012-06). Brussels, Belgium. Retrieved from http://pcp.vub.ac.be/papers/TaleofAdventure.pdf

Heylighen, F. (2012b). *Challenge Propagation: a new paradigm for modeling distributed intelligence* (No. 2012-01). Brussels, Belgium. Retrieved from http://pcp.vub.ac.be/papers/ChallengePropagation.pdf

Heylighen, F. (2013). Self-organization in Communicating Groups: the emergence of coordination, shared references and collective intelligence. In À. Massip-Bonet & A. Bastardas-Boada (Eds.), *Complexity Perspectives on Language, Communication and Society* (pp. 117–149). Berlin, Germany: Springer. Retrieved from http://pcp.vub.ac.be/Papers/Barcelona-LanguageSO.pdf

Heylighen, F. (2014). Return to Eden? Promises and Perils on the Road to a Global Superintelligence. In Ben Goertzel & T. Goertzel (Eds.), *The End of the Beginning: Life, Society and Economy on the Brink of the Singularity*. Retrieved from http://pespmc1.vub.ac.be/Papers/BrinkofSingularity.pdf

Heylighen, F., & Bollen, J. (1996). The World-Wide Web as a Super-Brain: from metaphor to model. In *Cybernetics and Systems' 96. R. Trappl (Ed.). Austrian Society For Cybernetics*.

Heylighen, F., & Bollen, J. (2002). Hebbian Algorithms for a Digital Library Recommendation System. In *International Conference on Parallel Processing* (p. 439). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/ICPPW.2002.1039763

Heylighen, F., & Chielens, K. (2008). Evolution of culture, memetics. In (R. A. Meyers, Ed.)*Encyclopedia of Complexity and Systems Science*. Springer. Retrieved from http://pespmc1.vub.ac.be/Papers/Memetics-Springer.pdf

Heylighen, F., Heath, M., & Van Overwalle, F. J. (2004). The Emergence of Distributed Cognition: a conceptual framework. In *Proceedings of collective intentionality IV*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.201.9282

Hill, R. A., & Dunbar, R. I. M. (2003). Social network size in humans. *Human Nature*, *14*(1), 53–72.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. MIT Press Cambridge, MA, USA.

Hutchins, E. (2000). Distributed cognition. In N. J. Smelser & P. B. Baltes (Eds.), *International Encyclopedia of the Social and Behavioral Sciences*. Elsevier Science.

Landauer, T. K., McNamara, D. S., Dennis, S. E., & Kintsch, W. E. (2007). *Handbook of latent semantic analysis.* Lawrence Erlbaum Associates Publishers. Retrieved from http://psycnet.apa.org/psycinfo/2007-04818-000

McLeod, P., Plunkett, K., & Rolls, E. T. (1998). *Introduction to connectionist modelling of cognitive processes*. Oxford University Press Oxford.

Miller, J. H., & Page, S. E. (2007). *Complex adaptive systems: an introduction to computational models of social life*. Princeton University Press.

Rodriguez, M. A. (2010). General-Purpose Computing on a Semantic Network Substrate. In Y. Badr, R. Chbeir, A. Abraham, & A.-E. Hassanien (Eds.), *Emergent Web Intelligence: Advanced Semantic Technologies* (pp. 57–102). London: Springer London. Retrieved from http://arxiv.org/abs/0704.3395

Rodriguez, M. A. (2012). Exploring Wikipedia with Gremlin Graph Traversals. *Marko A. Rodriguez*. Retrieved from http://markorodriguez.com/2012/03/07/exploring-wikipedia-with-gremlin-graph-traversals/

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing*. San Diego: University of California Press.

Valigiani, G., Biojout, R., Jamont, Y., Lutton, E., & Collet, P. (2005). Experimenting with a Real-Size Man-Hill to Optimize Pedagogical Paths. In *Proceedings of the 2005 ACM symposium on Applied computing* (pp. 4–8). Retrieved from http://apis.saclay.inria.fr/html/Papers/files/pdf/166_sac.pdf

Van Overwalle, F., & Heylighen, F. (2006). Talking nets: A multiagent connectionist approach to communication and trust between individuals. *Psychological Review*, *113*(3), 606.

Veitas, V. (2012). *Software architecture of the challenge propagation model* (GBI Working Paper No. 2012-06). Retrieved from http://pespmc1.vub.ac.be/GBI/vveitas.SoftwareArchitecture.pdf

Weinbaum, D. R. (2012). A Framework for Scalable Cognition: Propagation of challenges, towards the implementation of Global Brain models. *GBI working paper 2012-02*. Retrieved from http://pespmc1.vub.ac.be/ECCO/ECCO-Papers/Weaver-Attention.pdf

Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N., & Malone, T. W. (2010). Evidence for a Collective Intelligence Factor in the Performance of Human Groups. *Science*, *330*(6004), 686–688. doi:10.1126/science.1193147

Woolridge, M., & Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc. New York, NY, USA.