

A New Direction of Research into Artificial Intelligence

George Rzevski

Professor, Design and Complexity Science, The Open University, UK

Chairman, Rzevski Solutions Ltd, London, UK

Abstract: A new direction of research into Artificial Intelligence is outlined in this paper based on fundamentals of Complexity Science. Intelligence is postulated to be an emergent property of complex networks because it emerges from the interaction of network components and is not traceable to any of these components. Researching artificial intelligence is best conducted by designing artificial complex systems and tuning them to exhibit emergent intelligence.

Introduction

Research in Artificial Intelligence (AI) has a long tradition. The first paper attributed to the field was published by Warren McCulloch and Walter Pitts in 1943 [1], and the term “artificial intelligence” was proposed and agreed at the famous Dartmouth Workshop held in 1956. The thesis of this paper is that the direction of most of the past AI research was unnecessary biased towards logic and mathematics. It did produce some interesting results but not intelligent machines, as promised. A new and far more promising research direction is outline below and described in some detail in [2].

What is Intelligence?

The notion of human *Intelligence* is very complex; it comprises the following (and possibly many other) capabilities:

- **Understanding meaning** of symbols, words, text, data, images, utterances
- **Learning** (acquiring knowledge) from data, text, images as well as from own behaviour and behaviour of others and learning by discovery
- **Analysing** (deconstructing) complicated situations
- **Making choices** (decisions) under conditions of variety and uncertainty and therefore solving incompletely specified problems and achieving goals under conditions of the occurrence of frequent unpredictable events
- **Interacting** (communicating) with other actors in the environment, which include intelligent creatures and machines
- **Autonomously adapting** to changes in the environment
- **Creating** (constructing) new concepts, principles, theories, methods, artefacts, models, literature, art
- **Setting and achieving goals** by competing and/or cooperating with others

An important part of human intelligence is to strive to create **Artificial Intelligence**.

Artificial Intelligence

“Artificial” means man-made rather than natural. Artificial Intelligence is supposed to be man-made intelligence, designed and implemented in computer software and built into artefacts such as robots or intelligent machines [3]. Historically artificial intelligence

programs appeared in various disguises such as universal problem solvers [4], expert systems [5], [6] and neural networks [7].

The author proposes that *artificial intelligence should be considered as an emergent property of complex systems* [2]. A network of several billions of neurons in the human brain is such a complex system in which intelligence is not traceable to any individual component – it emerges from the interaction of these components (neurons).

It follows that *the most direct approach to creating AI is to construct artificial complex systems in software and to experimentally seek to obtain aspects of intelligence such as understanding of meaning, learning, autonomous adaptation and decision making under conditions of variety and uncertainty.*

We shall explore this thesis after reviewing the concept of Complexity.

What is Complexity?

The following three paragraphs from Wikipedia [8] are a good introduction to the concept of complexity.

“Complexity has always been a part of our environment, and therefore many scientific fields have dealt with complex systems and phenomena. Indeed, some would say that only what is somehow complex – what displays variation without being random – is worthy of interest.

The use of the term complex is often confused with the term complicated. To understand the differences, it is best to examine the roots of the two words. “Complicated” uses the Latin ending “plic” that means, “to fold” while “complex” uses the “plex” that means, “to weave.” Thus, a complicated structure is one that is folded with hidden facets and stuffed into a smaller space. On the other hand, a complex structure uses interwoven components that introduce mutual dependencies and produce more than a sum of the parts... This means that complex is the opposite of independent, while complicated is the opposite of simple.

While this has led some fields to come up with specific definitions of complexity, there is a more recent movement to regroup observations from different fields to study complexity in itself, whether it appears in anthills, human brains, or stock markets.”

Following the train of thoughts suggested above, the intuitive interpretation of the term Complex as “difficult to understand” is correct as long as we accept that the reason for the difficulty is *the interdependence of constituent components*.

An example that immediately comes to mind is the Internet-based Global Market, where consumers and suppliers are trading, each pursuing their own goals and targets, and where the overall distribution of resources to demands emerges from individual transactions rather than according to a given plan.

According to Prigogine [9] a system is complex if its global behaviour *emerges* from the interaction of local behaviours of its components (the system creates a new order). Prigogine in his writings emphasises that the behaviour of a complex system cannot be predicted and that, in general, the future is not given [10]; it is being created by actions of all those that participate in the working of the Universe. He discusses examples of complex systems from physics and chemistry, including molecules of air subjected to a

heat input, autocatalytic chemical processes and self-reproduction of cells. Emergent behaviour of complex systems is widely covered in literature [11] and applied to many domains, including economics [12].

To locate complex systems on a map of predictability, I proposed [13] the following system classification (see table below), in which complex systems are placed between random and stable systems.

CLASSES/ Features	RANDOM SYSTEMS	COMPLEX SYSTEMS	STABLE SYSTEMS	ALGORITHMIC SYSTEMS
Predictability	Total uncertainty	Considerable uncertainty	No uncertainty	No uncertainty
Behaviour	Random	Emergent	Planned	Deterministic
Norms of behaviour	Total freedom of behaviour	Some external guidance is essential	Governed by laws and regulations	Follows instructions
Degree of organisation	None	Self-organisation	Organised	Rigidly structured
Degree of control	None	Self-control by self-organisation	Centralised control	No need for control
Irreversible changes	Random changes	Co-evolves with environment	Small temporary deviations possible	None
Operating point	None	Operates far from equilibrium	Operates at an equilibrium	Operates according to the specification

Table 1. A classification of systems

The Key Elements of Complexity

Let us carefully examine the key elements of complexity emphasising those that are essential for the design of artificial complex systems [14].

1. Perhaps the most important feature of complex systems is that *decision-making is distributed* rather than centralised. Complex systems consist of interconnected autonomous decision making elements, often called Agents, capable of communicating with each other. There is no evidence of centralised control.
2. The autonomy of agents is not total. Every complex system has some global and/or local principles, rules, laws, or algorithms for agents to follow. The important point to remember is that agent's behaviour is never completely defined by these rules – they always have alternative possible local behaviours. In other words, complex systems always have a *variety* of possible behaviours and *uncertainty* which behaviour will be executed. The degree of freedom that is given to agents (decision makers) determines the system's ability to self-organise and evolve. When

uncertainty is insignificant, the system behaves predictably and lacks capabilities for self-organisation. When uncertainty is equal to 1, the system is chaotic (random). The adaptive complex systems operate “at the edge of chaos” or “far from equilibrium”. The occurrence of events that affect their behaviour is so frequent that there is no time for the system to return to its equilibrium.

3. Global behaviour of a complex system *emerges* from the interaction of constituent agents. However, because the decision-making freedom of agents is restricted, complex systems exhibit *patterns* of behaviour. Designers have a choice here. The degree of uncertainty can be adjusted to force system to follow specified broad patterns. The complete predictability should not be aimed for – it would prevent the system to self-organise and adapt, if and when required.
4. Complex systems are non-linear: the smallest external effects may cause large-scale shifts in system behaviour, the phenomenon known as *butterfly effect* (eg, as in climate systems) or as *self-acceleration* (eg, as in chain reaction in atomic explosions). Also, complex systems exhibit *autocatalytic* behaviour, that is, the ability to create new structures without any external help (eg, creation of organic structures from non-organic materials, under certain thermal conditions).
5. The distribution of decision-making implies interconnectedness of decision-making elements (agents). The links between agents can be strong or weak or nonexistent. The type of link between agents determines the responsiveness of the system when disturbed. Designers can weaken certain links between agents to reduce time required for ripples caused by a chain of changes to settle down.
6. The autonomy implies intelligence. Intelligence implies knowledge and a capability of applying knowledge to resolve uncertainty.

Technology for Constructing Models of Complexity

The most effective technology for constructing artificial complex systems, which exhibit all features described in the previous section, is multi-agent software [15].

In contrast to conventional software such as centralized schedulers, planners and optimizers, which from the start to the end follow algorithms, multi-agent software works primarily by exchanging messages: Intelligent Software Agents negotiate deals with each other, always consulting problem domain knowledge assembled in Ontology. Negotiations are conducted by a concurrent and asynchronous exchange of messages. The system is event-driven: it rapidly self-organises to accommodate events that affect its operation.

Problem domain knowledge is elicited and represented as a semantic network where concepts (classes of objects) are nodes and relations between concepts are links. Each object is characterised by attributes and rules guiding their behaviour. Such a conceptual knowledge repository is called *Ontology*.

A real-life problem situation is represented as a virtual network of instances of objects defined in Ontology and their relations. Such a problem description is called a *Scene*.

The elementary computational element is called *Agent*. An agent is a computer program capable of solving the problem at hand by consulting Ontology and using thus acquired knowledge to negotiate with other agents how to change the current Scene and turn it from the description of the problem into the description of a solution. Agents solve

problems in co-operation and/or competition with other agents. As *Events* (new orders, failures, delays) affecting the problem domain occur, agents amend the current scene to accommodate the event thus achieving *Adaptability*.

An agent is assigned to each object participating in the problem solving process (and represented in the scene) with a task of negotiating for its client (object) the best possible service conditions. For example, Passenger Agents and Seat Agents will negotiate takeoff/landing times and seat prices for requested air taxi flights. Closing a deal between a Passenger Agent and a Seat Agent indicates that a full, or at least partial, matching between *Demand* and an available *Resource* has been achieved. In case of a partial matching (eg, a passenger agrees to accept a later takeoff time but it is not pleased), his Agents may attempt to improve the deal if a new opportunity presents itself at a later stage (eg, if other passengers on the same flight agree an earlier takeoff time). The process continues as long as it is necessary to obtain full matches, or until the occurrence of the next event (say, a new request for a seat), which requires agents to re-consider previously agreed deals.

Agent negotiations are informed by domain knowledge from Ontology, which is far more comprehensive than “rules” found in conventional schedulers and normally includes expertise of practicing operators. Not all of this knowledge is rigid - certain constraints and if-then-else rules may be considered as recommendations and not as instructions and agents may be allowed to evaluate their effectiveness and decide if they should be used. In some cases agents send messages to users asking for approval to ignore ineffective rules or to stress nonessential constraints.

The power of agent-based modelling is particularly evident when the problem at hand contains a very large number of objects with a variety of different attributes; when there is a frequent occurrence of unpredictable events that affect the problem solving process; and when criteria for matching demands to resources are complex (eg, balancing risk, profits and level of services, which may differ for different participants).

As the process is incremental, a change of state of one agent may lead to changes of states of many other agents. As a result, at some unpredictable moment in time a spontaneous self-accelerated chain reaction of state changes may take place, and after a relatively short transient time the overall structure will switch its state practically completely. Once the resulting structure has settled, the incremental changes will continue.

It is evident from above discussions that agent-based software exhibits autonomy and emergent intelligence.

Architecture of Multi-Agent Software

A multi-agent software comprises the following key components: (a) Multi-Agent Engine, which provides runtime support for agents; (b) Virtual World, which is an environment where agents cooperate and compete with each other as they construct and modify the current scene; (c) Ontology, which contains conceptual problem domain knowledge network and (d) Interfaces.

How Multi-Agent Software Works

Software consists of a set of continuously functioning agents that may have contradictory or complimentary interests. Basic roles of agents, based on extended Contract Net protocol, are Demand and Supply roles: each agent is engaged in selling to other agents

its services or buying services it needs (Passenger Agents buy seats and Seat Agents sell them).

Current problem solution (current scene) is represented as a set of relations between agents, which describe the current matching of services; for example, a schedule is a network of passengers, seats, aircrafts and flights and relations between them.

The arrival of a new event into the system is triggered by the occurrence of a change in the external world; for example, when a passenger requests a seat on a particular flight, a Seat Request Event is triggered in the system.

The agent representing the object affected by the new event undertakes to find all affected agents and notify them of the event and the consequences (eg, the agent of the failed aircraft undertakes to find Passenger Agents linked to the failed flight and inform them that the flight is not available; the Aircraft Agent breaks the relevant relations and frees the Passenger Agents to look for other available flights).

The process of problem solving can run in parallel and asynchronously and, as a consequence, simultaneously by several active participants; for example, passengers that arrived at the website to book a flight simultaneously can all immediately start searching for suitable seats. All aircrafts assigned to flights can start immediately looking for free pilots. This feature is very effective because it eliminates a laborious building of flight schedules only to find out that pilots are not available for all selected flights.

The driving force in decision-making is often the presence of conflicts, which have to be exposed and settled by reconsidering previously agreed matches; for example, if a new flight finds out that the takeoff time slot it needs is already occupied, negotiations on conflict resolution start and, as a result, previously agreed flight-slot matches are adjusted (the takeoff time slot is moved to accommodate both flights) or broken (the time slot is freed). This capability to make local adjustments before introducing big changes is what makes agents-based problem solving so much more powerful in comparison with object-oriented or procedure-based methods.

A multi-agent system is in a perpetual state of processing - either reacting to the arrival of new events or improving the quality of previously agreed matches. The stable solution, when there are no agents that can improve their states and there are no new events, is hardly ever reached (agents are perpetually operating "far from equilibrium").

Solutions developed using multi-agent software fall into the class of open, non-linear and dissipative systems. As the number of relations increases in the system, the level of complexity of the resulting network goes up and, at a certain point, the need may arise to appoint additional agents to represent certain self-contained parts of the network whose nodes are already represented by agents. The increased complexity of solution structures may result in the creation of loops and the system may find itself in a local optimum. To avoid being stuck in a local optimum agents are from time to time given power to proactively seek alternative solutions. Attempts to avoid local minima are random (mutations).

Multi-agent systems can learn from experience as follows. Logs of agent negotiations are analysed with a view to discovering patterns linking individual agent decisions and successes/failures of the agent negotiation process. In future negotiations patterns leading to failures are avoided.

The pattern discovery process is itself agent based. An agent is assigned to each data element with a task of searching for similar data elements to form clusters. An agent is assigned to each new cluster with a task to attracting data elements that meet cluster membership criteria. Finally, clusters are represented as “if-then-else rules”.

Designing Artificial Complex Systems using Agent Technology

Systems that have been designed under my supervision or with my involvement, using the above principles are described in some detail in [16], [17], [18], [19], [20]. Advantages of adaptability in comparison with rigid systems, such as ERP (Enterprise Resource Planning), are described in a popular format in my paper entitled “ERP: Elephants Rarely Pirouette” [21].

The list is substantial and includes real-time, adaptive multi-agent systems for: managing 10% of world tanker capacity for global crude oil transportation (in use); managing 2000 taxis and other service vehicles in London (in use); managing an extensive road logistic system across the UK (in use); managing social entitlements of citizens in a very large region (in use); managing distribution of rental cars across Europe for a major global car rental organisation (successful trials; in the commissioning stage); managing a car manufacturing system (prototype); simulating virtual enterprises (prototype); managing document flow for a major insurance company (prototype); managing all business processes of a new aviation company (in the design stage); managing a catering supply chain (in the design stage).

To illustrate the power of agent-based adaptive systems let me outline complexity of the design problem that I am handling at present. The goal is to design an adaptive organisation based on teamwork, and a supporting intelligent multi-agent management system, which will make autonomously all operational decisions (how much to charge a customer for a flight, which pilot, aircraft, ground staff will be assigned to which duty, etc.) and manage domain knowledge required for strategic decisions (on expansion, on market penetration, on increasing business value) for a brand new enterprise. The enterprise network will enable rapid interaction of twelve multi-agent modules, including simulators, several schedulers, a demand forecasting system, a human resource management system, and customer relations management system, and will maintain integrity of all enterprise data and several enterprise ontology. The system is being designed to handle 4,000 travel requests a day, to book 400 taxi seats/flights a day and to schedule or re-schedule a large fleet of small aircraft, flights, crews, ground staff, aircraft maintenance, fuel supply, etc every 7 seconds, which is a task which would be impossible to achieve without agent-based technology.

I have also researched, simulated or prototyped a number of distributed, adaptive *products* following the design principles outlined above, including: a machine tool; an intelligent geometry compressor, an autonomous parcel distribution system and an intelligent family of robots [22], [23].

Perhaps the boldest idea was to design a compressor with moving individual vanes capable of autonomously and dynamically positioning themselves at the optimum angle whenever the operating point of the compressor changes. A software agent is assigned to each individual moving vane equipped with a pressure sensor. As pressure on the vane changes, the Vane Agent negotiates with agents of other vanes how to change vane angles to achieve the optimum pressure distribution along the stator. Before making a decision, agents consult domain knowledge stored in individual agents’ minds, which can be updated without interrupting work of agents, to fine tune compressor operation. A very

successful simulation [24] showed that the compressor with autonomous vanes, when coupled with an aircraft jet engine, is fully adaptive to sudden changes of loads and is able to prevent stalling of jets caused by lack of air intake.

Replacing a robot by a family of smaller robots illustrates the advantages of designing complexity into artefacts even better. To avoid disasters that ruined both American and British Mars exploration robots (the first died from the accumulation of space dust on its solar cells after a week in space, and the latter fell into a crevice on landing and was immediately lost), I proposed to design a family of five smaller robots capable of cleaning each other, rescuing members of the family from disasters and, more importantly, able to complete their task successfully even if one or two of the family members were disabled.

A family of robots is an adaptive distributed system, which incorporates all critical complexity features listed earlier in this paper. All decisions are executed after a process of consultation and negotiation among members of the family. There is no “senior” robot ordering others what to do. Each robot is controlled by a set of interacting swarms of agents. Agents consult domain knowledge before making decisions. A copy of domain knowledge is stored in each robot’s ontology, making it capable of undertaking any task within domain boundaries. The family represents a “swarm of interacting swarms” of agents and therefore exhibits a considerable emergent intelligence. Robots are trained to help each other, share the workload and self-organise the team if a member is disabled without losing ability to achieve the goal.

Experimenting with Multi-Agent Systems

Using large-scale complex systems based on agent technology for research purposes is relatively straight forward if one stores all messages exchanged among agents. By shifting through the log of messages one can find connections between agent decisions and particular behaviours of the system and thus can deduce conditions under which desired behaviours emerged. The process is laborious but can be automated using appropriate agent-based software. My interest was in isolating system behaviours that could be described as aspects of emergent intelligence (understanding of meaning, learning, analysis, decision making under conditions of uncertainty, adaptation and autonomous creation of novel structures) as described in [2].

Conclusions

Constructing artificial complex systems using multi-agent technology and conducting experiments aimed at provoking a complex software system to exhibit emergent intelligence is a new approach to studying AI. The pre-requisite skills are skills of designing complexity into software, a notion that appears to be counter-intuitive. The conventional wisdom is to ensure that software is rigidly structured and “correct”. The new thinking is to make software autonomous, adaptable and self-organising (and therefore unpredictable), in other words, intelligent.

References

1. McCulloch, W.S. and Pitts, W. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. *Bulletin of Mathematical Biophysics*, Volume 5, 1943, pp 115-137.
2. Rzevski, G., Skobelev, P., “Emergent Intelligence in Large Scale Multi-Agent Systems”. *International Journal of Education and Information Technology*, Issue 2, Volume 1, 2007, pp 64-71.
3. Rzevski, G (ed), “Mechatronics: Designing Intelligent Machines”, Butterworth Heinemann, 1995.

4. Newell, A. and Simon, H. A. GPS, a Program that Simulates Human Thoughts. In Billing, H. (ed), Lernende Automaten, 1961, pp 109-124.
5. Feigenbaum, E. A., Buchanan, B. G. and Lederberg, J. "On Generality and Problem Solving: A Case study using the DENDRAL Program". In Meltzer, B. and Michie, D. (eds) Machine Intelligence, Volume 6, 1971, pp 165-190.
6. McDermot, J. "R1: A Rule-Based Configurer of Computer Systems". Artificial Intelligence, Volume 19(1), 1982, pp 39-88.
7. Rumelhart, D. E. and McClelland, J. L. (eds) "Parallel Distributed Processing" 1986, MIT Press, Cambridge.
8. Wikipedia www.wikipedia.com
9. Prigogine, Ilya, "The End of Certainty: Time, Chaos and the new Laws of Nature". Free Press, 1997.
10. Prigogine, Ilya, "Is Future Given?" World Scientific Publishing Co., 2003.
11. Holland, John, "Emergence: from Chaos to Order". Oxford University Press, 1998.
12. Beinhocker, Eric, "The Origin of Wealth: Evolution, Complexity and the Radical Remaking of Economics".
13. Rzevski, G., "Investigating Current Social, Economic and Educational Issues using Framework and Tools of Complexity Science". Journal of the World University Forum, Volume 1, Number 2, 2008.
14. Rzevski, G., Skobelev, P., "Emergent Intelligence in Large Scale Multi-Agent Systems". International Journal of Education and Information Technology, Issue 2, Volume 1, 2007, pp 64-71.
15. Rzevski, G., Skobelev, P. Andreev, V. "MagentaToolkit: A Set of Multi-Agent Tools for Developing Adaptive Real-Time Applications". In Marik, V., Vyatkin, V., Colombo, A. W. (eds.) Holonic and Multi-Agent Systems for Manufacturing. Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2007, Regensburg, Germany, September 2007, pp 303-314. Springer LNAI 4659.
16. Rzevski, G., Himoff, J., Skobelev, P., "Magenta Technology: A Family of Multi-Agent Intelligent Schedulers". Workshop on Software Agents in Information Systems and Industrial Applications (SAISIA). February 2006. Fraunhofer IITB.
17. Andreev, M., Rzevski, G., Skobelev, P., Shveykin, P., Tsarev, A., Tugashev, A. "Adaptive Planning for Supply Chain Networks". In Marik, V., Vyatkin, V., Colombo, A. W. (eds.) Holonic and Multi-Agent Systems for Manufacturing. Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2007, Regensburg, Germany, September 2007, pp 215-225. Springer LNAI 4659.
18. Rzevski, G, Skobelev, P, Batishchev, S, Orlov, A.: "A Framework for Multi-Agent Modelling of Virtual Organisations". In Camarinha-Matos, L M and Afsarmanesh, H (eds), Processes and foundations for Virtual Organisations, Kluwer Academic Publishers, 2003, pp. 253-260.
19. Minakov, I., Rzevski, G., Skobelev, P. and Volman, S., "Creating Contract Templates for Car Insurance Using Multi-Agent Based Text Understanding and Clustering". In Marik, V., Vyatkin, V., Colombo, A. W. (eds.) Holonic and Multi-Agent Systems for Manufacturing. Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2007, Regensburg, Germany, September 2007, pp 361-371. Springer LNAI 4659.
20. Rzevski, G., Skobelev, P., Minakov, I. and Volman, S., "Dynamic Pattern Discovery using Multi-Agent Technology". Proceedings of the 6th WSEAS International Conference on Telecommunications and Informatics (TELE_INFO '07), Dallas, Texas, USA, March 22-24, 2007, pp 75-81. ISBN: 978-960-8457-60-7.

21. Brace, G, Rzevski, G, ERP – “Elephants Rarely Pirouette”. Logistics Focus Volume 6, No 9 1998.
22. Rzevski, G.: “On conceptual Design of Intelligent Mechatronic Systems”. Mechatronics 13 (2003) pp. 1029 – 1044.
23. Rzevski, G, “Engineering Design for the Next Millennium: The Challenge of Artificial Intelligence”. The 86th Thomas Hawksley Memorial Lecture, IMechE, 9 December 1998.
24. Morgan, G, Rzevski, G., Wiese, P.: “Multi-Agent Control of Variable Geometry Axial Turbo Compressors”. Journal of Systems and Control Engineering, issue I3 vol. 218 (2004), pp. 157-171.