THE SOCIAL COMPUTER – COMBINING MACHINE AND HUMAN COMPUTATION

Fausto Giunchiglia and Dave Robertson

May 2010

Technical Report # DISI-10-036

The Social Computer – Combining machine and human computation [1]

Fausto Giunchiglia
Dept. of Computer Science and Information Engineering
University of Trento
38050 Povo, Trento, Italy
fausto@disi.unitn.it

and

Dave Robertson
School of Informatics
University of Edinburgh
Informatics Forum, Crichton Street
Edinburgh EH8 9AB, UK
dr@inf.ed.ac.uk

Abstract. The *social computer* is a future computational system that harnesses the innate problem solving, action and information gathering powers of humans and the environments in which they live in order to tackle large scale *social problems* that are beyond our current capabilities. The *hardware* of a social computer is supplied by people's brains and bodies, the environment where they live, including artifacts, e.g., buildings and roads, sensors into the environment, networks and computers; while the *software* is the people's minds, laws, organizational and social rules, social conventions, and computer software. Similarly to what happens within a conventional computer and more interestingly within naturally occurring reasoning and control systems like the human body, a social computer exhibits an algorithmic behavior and problem solving capabilities which are the result of very large numbers of local computations, decisions, interactions, data and control information transfers.

1. The vision

The *social computer* is a future computational system that harnesses the innate problem solving, action and information gathering powers of

[1] The elaboration of these ideas started when the first author gave the talk "The future of AI", at the Symposium in honor of Alan Bundy, Edinburgh, July 2008 [4]. Further elaborations have come out of many discussions during the preparation and development of the FET FP7 IP "Forever yours" project "Living Knowledge" (grant agreement n° 231126) [7]. The Trentino flooding explanatory example is an elaboration of the results of the STREP FP6 Open Knowledge project [8] which was also the source of many seminal ideas.

humans and the environments in which they live in order to tackle large scale *social problems* that are beyond our current capabilities. Examples of such problems are: a large scale fast reaction to a big global threat (e.g., a tsunami or a flooding), a large scale slow reaction to a global threat (e.g., the 2009 financial crisis), a large scale reaction to a local problem (e.g., the need to acquire the otherwise not locally available knowledge needed to identify a very rare disease), the goal of reaching a global objective (e.g., how to save energy via the coordination of distributed production and consumption units), and so on.

The *hardware* of a social computer is supplied by people's brains and bodies, the environment where they live, including artifacts, e.g., buildings and roads, sensors into the environment, networks and computers; while the *software* is the people's minds, laws, organizational and social rules, social conventions, and computer software. The *algorithms* for social computation are those which take advantage of the resilience and scale of mass human problem solving and which achieve higher performance by supplementing social capabilities of individuals by adapting the environment in order to provide better feedback and instrumentation for the mass computation achieved by society. This means that the analogue of an operating system for a social computer is not just the computer or the network but is primarily the social and environmental infrastructure that enables large-scale social computation to occur.

Similarly to what happens within a conventional computer and, more interestingly, within the human body, a social computer is associated with an *identity* and a *boundary* which defines what is inside (its many parts) and what is outside (the environment where the computer lives). Inside and outside interact via computer networks and real world physical or human enabled communication. A social computer is characterized by being distributed but *limited* over a collection of computational elements (people, organizations, system) and, in certain cases, also in space, for instance within a region or a town. The limitation in extent defines its identity and allows it to act within a limited amount of time, for instance in order to react to a given stimulus, or to achieve a goal. Social computations also effect the state of the physical environment; as a consequence a social computer must be capable of making its components interact and exchange data, information and knowledge, but also to transfer them from one place to another (e.g., people, cars, goods may be moved from one building to another building). This requires networked infrastructure that connects the various sensors, actuators and actors in the environment and, equally importantly, allows fast and reliable transfer and transformation of information flowing through this network.

The social computer exhibits an *algorithmic behavior* which is however the result of very large numbers of local computations, decisions, interactions, data and control information transfers. These activities start, evolve and stop simultaneously inside the computer's many parts and across them, with its components being most often completely unaware and sometimes partially aware of the resulting global algorithm behavior. The *state* of a social computation includes part of the state of the physical, social environment in which the computation is situated.

The algorithmic behavior of the social computer can be characterized at two different levels of abstraction:

1. A *inner level* (or *loop*): the level of what is nowadays called *social computing* [2], where computers and people collaborate in a distributed way to produce some global emergent behavior;
2. An *outer level* (or *loop*): This is the level where the global behaviors produced by social computing are composed and integrated towards solving the large scale social problems mentioned above.

Inner and outer level can be iterated to produce more and more abstract and complex behaviors as in "usual" programming. The key observation is that the algorithms run at the outer level(s) produce *consistent and persistent problem solving* performance, despite the fact that part of this behavior is emergent (in the sense that it is derived from the individual intelligences within a society). Thus, for instance, in the case of tsunami response, the social computer might have as a goal to minimize the number of casualties and the ideal algorithm for social computation in this domain would harness the natural human instinct to minimize personal injury in emergency situations in order to obtain consistent minimization of casualties overall. Individual reactions to the ongoing computation should tend to reinforce the algorithm so that, as more people participate, its performance improves.

The problem solving capability of the social computer *improves in time.* This result can be achieved because it is *programmable,* and the set of programs that it can run can be increased in time, in quantity and quality. In this context the notion of being programmable is very different sense from that which applies to usual computers, even those that are distributed. For instance, people cannot be programmed; yet they can be induced into certain behaviors via, e.g., incentives or laws. And there are already examples where people are used as computing entities whose intelligence and activities are composed to achieve a more complex goal [3,9,10]. The behavior of a social computer is programmed by increasing and evolving the amount of globally available *knowledge about the world*, and about itself, and its capability to make maximal *pragmatic use* of it. Notice that here we use the term knowledge in a very broad sense, including data of all the various kinds (structured, semi-structured, unstructured, text and media of various kinds), information, structured knowledge, but also specifications of processes and services.

The social computer can be programmed thanks to its *massive local learning* capabilities (reaching into human society and the social environment) which, in turn, rely on the following set of requirements:

- The ability of the social computer to *store* a virtually unbound amount of knowledge in the form of general knowledge and, more importantly, of concrete ground knowledge about its components. Some of this knowledge is also generated in *real time*, for instance via sensors in the environment or social networking among people. This global knowledge is the *union* of the *local* knowledge of its components. The local knowledge of each and any component is usually highly *diverse* from that of the others and, consequently, is partial and often mutually inconsistent (where we use the word "diversity" with the meaning defined in [1]).
- The ability for each and any component of the social computer of finding and *retrieving* the needed specific knowledge in real time, usually from other components.

- The ability of *transferring* the needed quantity of needed knowledge from/to any two components in real time.
- The ability for each and any component of *using* the knowledge transferred from other components despite the inherent diversity and consequent difficulties.

Last but not least, the computer can *sense* and *act* in the world. People are one the main means for sensing but this can also happen via pervasive sensors and sensor networks. People are the main actuators of the social computer. The social computer can also act via machines but ultimately, modulo limited exceptions (e.g., intelligent machines, automatic control devices) the final decision on how to act remains a human decision, possibly supported by advanced decision support systems. One could easily envisage scenarios involving social robots autono-mously acting in the real world. While this is a research area of po-tential, we believe that the current state of the art does not suggest that we consider robots as key components of the social computer in the medium term.

2. A metaphor and an example

As from the previous section, the vision is that a social computer is a complex object that is able to produce globally coherent behavior based on the composition of many simpler behaviors contributing to its realization, while being unaware of "bigger picture". The proposed artifact, though quite innovative, can be compared with the human mind and its "hardware realization" in terms of the human brain and body, the latter being composed from very complex objects that, nevertheless, are able to produce behaviors satisfying the properties listed in the previous section. The early cognitive science literature [14,15] and various attempts to build a computational (artificial) intelligence [16,17] have hitherto had the effect of polarizing these two visions, now made complementary through advances in technology and modern society. We believe that the same will happen with the social computer.

With this in mind, we provide an example of social computer by paralleling it with the behavior of a human, hoping that this exemplifies (rather than complicating) our vision. Section 7 will further exploit this parallel to motivate the plausibility of the ideas presented.

Let us think of a person, Alice, walking in a street. All the sudden, unexpectedly, a car gets very close to her and she reacts very rapidly and jumps on the sidewalk. What can we notice in this small example?

- Alice has a clear, bounded spatial extension: her body and its periphery.
- Alice's behavior can be said to be algorithmic and she can perform a problem solving activity with a precise goal to be achieved: avoid being hit by the car. At the same time probably none of her "components"; e.g. her legs, arms, muscles and nervous system, are directly aware of the overall goal to be achieved.
- Alice's problem solving capability is consistent and replicable: if and when a similar situation arises she will again try to avoid the car. The overall high level behavior will be the same even if many of her "components" may turn out to have a locally different behavior. Thus for instance one of her arms may end up being above her head or waving in the air. However the specific final state of her arm is not closely relevant to the goal to be achieved. The action may be successful or fail, as a function of the local state (e.g., the ability of Alice to perform a long enough jump).
- Alice's problem solving capability can be improved over time: she will remember this experience, she will learn from it (e.g., avoid walking in the middle of the road), she will be able to integrate it with the previous knowledge and reuse it in future situations. Even more than this, by suitable training, she will be able to improve her skills and knowledge and thus to perform much better in a similar future situation.
- Alice bases this behavior and its adaption on signals from her own senses and actuators.

Let us now consider Trentino, a small region in the North of Italy with full broadband, optical and wireless network covering all its territory and connecting its approximately 600,000 inhabitants. As periodically occurs in the region, imagine that in 2020 there is large scale flooding whose risk was anticipated only a few hours before the event itself. The Trentino social computer reacts immediately with the overall goal to save as many people a possible and to minimize the overall damage. In particular, reiterating the analysis of the previous example:

- The Trentino social computer has clear geographical boundaries which correspond to the territory of the Trentino region.
- The behavior of the Trentino social computer can be said to be algorithmic and to have a problem solving capability with a precise high level goal to be achieved. At the same time, most (if not all) its components are unaware of the overall goal (operationally stated). A few people (e.g., the police, the emergency handling people, the health operators) have a global societal goal, which is anyhow only part (a sub-goal) of the overall goal, while most of the population is concentrated in their local goal and artifacts, including computers, are simply passive or do what they are instructed to do.
- The problem solving capability of the Trentino social computer is consistent and replicable even though there will be many local results (e.g., specific cars (not) being destroyed by the flooding) which are different in different floodings.
- The Trentino social computer can improve over time: it will remember this experience, and future training and preparation will make sure that next time the negative effects of the flooding will be even less.
- The Trentino social computer uses all its own sensors and actuators while reacting to the flooding.

But where is the novelty? Isn't this what we already do now? Yes and no. The difference is in the use of knowledge in order to make the social problem solving process (which is quite chaotic in nature) algorithmic, replicable, of increasing quality, where *the high level societal goal is effectively and efficiently instantiated into the required millions of lower level societal sub-goals, down to organizational and individual sub-goals.* And this is exactly where ICT, its current pervasiveness of being "*anytime, anywhere and for anybody*" can make the difference. We list below some concrete examples, expanding on the Trentino scenario, of pervasive systems that feed social computation:

- All the people, via their cell phones, could be kept informed of the evolution of the situation, e.g., where the flooding is going, where buses or boats are, where the convergence points are, and so on.
- A specific piece of medical knowledge could be brought to an untrained person needing it from the Hospital knowledge base in a form which is easily understandable.
- A doctor could be found, via the phone directory, and driven to a specific place in urgent need of her knowledge. The path to the place could be provided to the doctor on her cell phone on the basis of the real time information of the direction of the flooding. In parallel, a dedicated program could establish from a hospital database that the doctor does not have the right background and the relevant information from the previous item could be brought also to her cell phone.
- But how was it possible to understand what was the right "piece" of medical knowledge and that a doctor was needed? This could occur via a phone call from the person to the police and from the ability of a program, via appropriate data access, to immediately put the policeman in contact with an available person in the Hospital call center.
- In parallel, the first available and closest ambulance (taken from a Red Cross data base) could be brought as close as possible to the place where the injured person who is then driven to the best possible hospital (in terms of being geographically close, of having available rooms, and the best possible competences).
- … and so on, for all the possible small and large situations and problems to be solved, involving one or more of the many components of the social computer.

3. The social computer and system architectures

We believe that there is an interesting and overall convergence in software and knowledge architecture that makes it timely to build a social computing architecture. In the internet world, there has been a strong shift towards service-oriented architectures based on the notion of encapsulating significant functionalities as services and making these available to other services as a way of producing larger systems via components with trusted interfaces [11]. In the mobile device world we are seeing services cluster around particular types of data provision and design of components being standardized in order to allow consistency of service over time, based on strong standardization on the means of coordinating those services [12]. In the world of physical, robotic devices we have seen behavior-based architectures dominate, in which the modularity of "service" is defined in terms of a behavioral competence that can be trusted to a given tolerance as a component of other, more complex behaviors [13]. In the knowledge world diversity has been recognized as a feature which must be exploited [1,7] thus leading the way towards the integration of isolated and diverse pieces of knowledge. The "ICT forever yours" call [6] and the underlying core emphasis on diversity are further evidence of this phenomenon.

In all of these areas, the core idea (even if not articulated in this way) is that *diversity* must be handled based on an understanding of the knowledge to be shared and the program design metaphor should be that of module definition accompanied by coordination of modules to obtain aggregate behaviors that are evolved through use. The scale of the system means that different modules and coordination activities are developed independently, with different local goals and implicit assumptions, and they evolve over time. Time adds a further dimension to the problem as modules and coordination activities developed at different times will reflect the evolution of society [7].

In a social computer, knowledge and services are provided by society and the performance of coordinated problem solving is evaluated by that society, so that the design and evaluation phases within the architecture follow a virtuous cycle. However for this to happen two fundamental requirements must be satisfied:

1. there must be a mapping between the software (data and programs) modules and their interactions and the corresponding social components and interactions;
2. the pervasive social diversity must be mapped into the corresponding pervasive diversity in software.

It is important to notice that the software diversity and modules are a mirror of the social diversity and "modules", the issue is how to formalize this mirror so that society and programs keep aligned. This requires *mapping the architecture of the social computer into the organizational structure of society*. Although these are very early days and devising such an architecture will be one of the biggest challenges, we believe that a set of patterns will emerge simply as a consequence of the need for computation at large scale to resemble society and its organization. Thus, for instance, the social computer will most likely be organized as a set of functional *apparati*, most often organized hierarchically, which implement a well defined function, e.g., emergency response, health, energy management, governance, the infrastructure for mobility and storage (e.g., buildings) and so on.[2] Similarly, these apparati will most likely work autonomously and largely *independently* of one another and will synchronize and exchange knowledge in a relatively small number of very specific and localized ICT enabled bridge apparati.

A small concrete instantiation of the general architecture proposed above on the examples in Section 4 leads to the following instantiations. In the first example the apparati involved will be the respiratory apparatus, the cardiovascular apparatus, together with the nervous and the motor systems. In terms these human apparati can be decomposed in simpler apparati (e.g., the lungs and the heart) and interact in very precise locations, that is cells, and there are different cells with different functionalities for different apparati. Thus for instance, the vascular system and the motor system interact in the muscle cells. In the second example the apparati involved are the Trentino health system, the telecommunication system, the emergency response system, the local communities in parts of Trentino, and so on. The bridge apparati perform both data homogenization and process synchronization.

2 We expect that a set of initial results in this direction will be achieved as part of the efforts of the "Future Internet (FI) Initiative". Thus for instance, as part of the FI initiative, it is possible to identify research and innovation areas such as eMobility, eHealth, eInclusion, eEnergy, eGovernance, and so on.

4. Research issues

Although the structural conventions for social computation are familiar (knowledge sharing, hierarchical organization of components, trusted components, orchestration via mediation) the actual process of design is radically different. Our goal is to enable social computation directly to meet key societal challenges by improving the quality of decision making, targeting, and timeliness of response to need. The social computer will empower us to configure information, human, computational, robotic and environmental resources to create new hybrid structures to respond to challenges in the 21st century. This requires new approaches to architecture that acknowledge the power of an open, transparent, structure that stimulates innovation while recognizing the need for democratic control that protects the rights of individuals, minority groups and property owners. The architects of social computation need a skill set that draws deeply on *interdisciplinary research* into humans, organizations and society:

- *Cognitive science*, researching the match between human cognitive capacities and the new environment enabled by the social computer.
- *Economics*, researching quantitative aspects of the transformation brought about by the social computer, including incentives to use social computation, economic impact, the effects of competition, the stability of new markets and mechanisms.
- *Sociology* and *organizational science*, researching the synergy between the new technologies underpinning the social computer and the existing social and organizational structures together with the trajectory of adoption and successive adaptations and evolution of social structures and technologies as the social computer diffuses through our societies.
- *Law*, which will provide us with the means for dealing with legislative consequences of the approach and its negative effects (attacks, viruses and other threats to the social computer).
- *Criminology*, researching the evolution of deviancy, conflict and crime in the new environments enabled by the social computer.
- *Ethics,* will provide us with the foundations for the creation of a better society.
- *Innovation, Technological innovation, Service innovation, Social innovation,* which will provide us with the required knowledge of

the difficulties encountered when trying to transfer technology to the real world.

When proposing a new research line as disruptive as the one proposed here, an important question is whether this will stimulate innovation, regardless of whether its ultimate goal is reached. This question is also particularly important as Europe does very well in terms of research production but much less when the issue becomes how to transfer the research results into (technological) innovation. We believe that the development of the social computer will generate a big improvement in innovation capability, no matter how far we travel in its development. As we have discussed elsewhere [5], we believe that the current best approach to innovation (in particular in Europe) is to do technology pull putting the *end user at the center of the innovation process*. This can be done by designing *service innovation* in a way that enables *technological innovation* and this is exactly what the social computer does by putting society (and the individuals acting inside it) at the core of the computation process. In this perspective social innovation can again be put at the core of the social computer via the development of new socially aware software development methodologies.

In turn, the studies developed by the disciplines listed above will provide input to ICT. Virtually all the ICT sub-disciplines will be impacted, most noticeably: data and knowledge management, service-oriented architectures, software engineering, privacy and security, and, ultimately *all the Web and Internet focused disciplines*. A first list of research issues that will have to be addressed, includes:

– Defining a notion of computation which embraces society as well as the mechanics of traditional computation and also harnesses social scale to accommodate local behavioral deviations.
– Designing and developing much more abstract design methodologies and languages, exploiting notions which are much closer to the societal notions (e.g., notions such as goal, actor, plan, activity, community, group) and where computers and humans are treated uniformly.
– Designing and developing the "society based" technology which must enable the storage, indexing and retrieval, transfer and use of the social computer knowledge (data and programs).

- Charting the space of possible architectures for social computer based systems.
- Handling knowledge diversity of data and programs in space and time and, consequently, adaptivity, mutation, and evolution.
- Building run-time environments for monitoring, diagnosis, compensation.
- … and so on.

The development of the social computer will require not only *engineering*, e.g. the construction of new prototypes aimed at solving specific problems, but also fundamental *science* aimed at explaining the emerging phenomena (some of which are already arising). This will require experimentation and, more in general, an empirical approach which will provide the basis for the development of the theoretical foundations of the social computer.

The development of the social computer will lead to a radical departure from the current practice in ICT and also in the other disciplines (think for instance of the design or experimentation processes, which will have to be deeply aware of the social and economical notions which will underlie the behavior of the social computer). We believe that future ICT will go beyond the differences and barriers which now exist among the ICT sub-disciplines and the other disciplines. The integration between ICT and the other disciplines will not see ICT as a simple instrument, nor as a cause of a simple change of approach. We see a *process of bi-directional, interdisciplinary mutual convergence which will change all the disciplines, including ICT.* In general, the development of the social computer will require a holistic approach where which will require an in depth merge of the existing know-how which, in turn, will generate new disciplines which will position themselves at the boundaries of the current disciplines.

5. How social computation changes the way we solve problems and build algorithms

To demonstrate the change in approach necessary to build large scale social computation systems, let us start with a (seemingly) simple problem. Imagine that everyone in some geographical region has the ability to supply observational data about some new disease that is sweeping through the area and has a virulent form (d1) and a non-virulent form (d2). People reporting a specific symptom s1 would like to know whether this means they have d1 or d2. Opinions in the medical profession differ, however, over the relationship between symptoms and disease: one school of thought says (1) that s1, plus other symptoms, is evidence for d1; another says (2) that s1, plus other symptoms, is evidence for d2.

How do we solve this problem in the classical style?

1. *Understand the problem, then deploy*: Go back to the medical profession and obtain consensus. Let's say that option (1) eventually obtains medical backing. Then we make the diagnostic algorithm for option (1) and make it available as a service for our population, each individual of which now has 100% chance of getting the correct diagnosis. This approach succeeds only in the unlikely event that lasting consensus can be reached with certainty in a timeframe appropriate to the (in this case quite pressing) social need.
2. *Data-intensive analysis, then deploy*: More commonly, the selection between options (1) and (2) is made by testing the performance of each against what we observe in the population, so data analysts study what happens when part of the population develops s1 and use that as evidence before choosing, say, option (1) and making it available as a service. This, however, leaves the population without a service until there is enough evidence to choose option (1). It is therefore unlikely to succeed in cases where the disease moves through a population rapidly – we end up with a 100% successful service after the need for it has passed.
3. *Deploy all relevant solutions*: Make services based on option (1) and option (2) available right away and let people use them. Assuming that both services are equally accessed and that s1 is indeed indicative of one or other of d1 or d2, 50% of the population will receive the right advice. This is a poor success rate (no better than flipping a coin) but in many practical situations may be better than the other options above which make unrealistic assumptions about the problem in order to strive for a 100% successful service on deployment.

Let us now contrast this with a social computation for the same problem. This proceeds as follows:

1. *Deploy all relevant solutions*: Identical to strategy 3 above. Initially we have a system with a 50% success rate.
2. *Individuals give local feedback*: Give users of services a means to rate the effectiveness (1) and (2) based on personal, individual experience of actually developing d1 or d2 and comparing that to the advice given by the service used. If option (1) is the correct one then its rating will increase at the expense of option (2).
3. *Individuals inform collective choice*: Give users of services a means of discovering the ratings of others. If the ratings are accurate then this will influence more of the population to choose option (1) and the success rate overall will be greater than 50%, approaching 100% as the ratings identify a clear winner.
4. *Collective choice influences service provision*: Give service providers incentives based on social feedback. In our case, the service based on option (2) would fall into disuse, unless it could be reconfigured to compete for ratings against option (1).

This social computation has several advantages over the classical approaches given above. It starts with at least some success (50%, whereas classical options 1 and 2 stay at 0% for a disturbingly long time). It is likely over time to increase its success rate, and do so as an integral part of operating within the society it serves. Crucially, this increase in success rate in steps a-c above is due to society itself, not to the conventional computations relating s1 to d1 or d2. In step d, this is taken a stage further by adding a social influence to the design of the classical computations.

Algorithms for social computation can incorporate conventional algorithms but a specific set of characteristics equip an algorithm for social computation. These characteristics are themselves social, rather than purely abstract properties:

– *Social reinforcement through local incentives*: At the core of the algorithm there must be a self-reinforcing system of feedback such that the incentive for an individual to supply data to the algorithm increases as more individuals participate. In our example above the incentive for individuals to use the social system might be that by supplying ratings their own diagnosis as well as those of their peers would be likely to improve.

- *Scaling to society*: The performance of the algorithm should increase as more individuals participate. In our example above, the mechanism for propagating ratings and using this to influence choice of diagnostic service would have to do little more than statistically measure the actual accuracy of diagnosis experienced by individuals – a lightweight process that scales to large populations but one which harnesses a complex human process (each individual's judgement).
- *Correctness as a social measure*: Whether or not the algorithm gives the right result is determined by the aggregation of experience built into the algorithm itself. In our example above, the correct diagnostic answer is estimated by those using the algorithm and the estimate is fed back into the behavior of the algorithm.
- *Completeness as a social judgement*: Since social algorithms begin as incomplete problem statements and grow to cover more of the problem via interaction with the population, there is not necessarily any specific point at which we can insist that we have a complete specification of either the problem or the algorithm to solve it, particularly since the environment within which society operates is subject to change. In our example above, disease d2 which was non-virulent might become virulent (or vice versa for d1) so a simple notion of completeness cannot usefully be applied to this system – it is much more useful to ensure that the system converges on whatever partial completeness is available.
- *Evolution through social influence*: A social algorithm must evolve with society because the effectiveness of the algorithm depends on its links from and to that society. However, initial algorithms are likely to be imperfect and use imperfect components so the design of the algorithm must include incentives for those maintaining system components to improve them in response to social pressure. In our example above, there is no reason to have only two services offering diagnoses for d1 and d2; we could have many services and allow the rating system to direct individuals at those with greater predictive power, thus driving system evolution through a market conditioned by the algorithm itself.

Social algorithms are still algorithms in the traditional sense but the art of building such algorithms lies in allowing society (in the physical world) to take much of the burden of dealing with complexity of problem solving, building lightweight but scalable algorithms (in the virtual world) to pull data in from individuals; generate new information of higher utility to individuals based on the social interaction; and return the higher utility information to individuals in such a way as to reinforce their participation in the algorithm.

6. Example applications

In Section 5 we explained how the architecture of a social computer can change fundamentally the "rules of the game" of traditional, large scale computational problems in target areas. The following are examples of the way in which a traditional hard problem is radically altered by viewing it as a social computation.

Maintaining resilience in adverse conditions under time pressure. The standard means of controlling a public emergency (such as an evacuation during flooding or re-routing of traffic in response to damage of road networks) is to develop a plan for dealing with the emergency; fund a central agency that coordinates the plan; then coordinate activities through that agency during plan enactment. This way of operating is weak against damage (either directly to the central agency or indirectly through breakdown in communication channels between the central agency and those "on the ground"). A social computation solution allows those on the ground to communicate via personal devices enabled with shareable, though localised, plans of action and cooperation so that information about the unfolding emergency can propagate without routing through a central agency. This gives local information immediate value and substitutes an opportunistic local trading system (with the incentive for supplying information being access to immediate information from others who understand different parts of the overall emergency) for a centralised authority. Although the localised plans for coordination are fundamentally different from those that might have been devised by the central agency, the cumulative effect of their enactment by may be as good or better than centralised plans.

Maintaining resilience in adverse continuously changing conditions in time. There is a growing need of controlling and possibly decreasing energy consumption. In parallel, due to the production of increasing quantities of alternative energy, there is growing number of energy providers, possibly for small quantities, distributed geographically (down to the single producer in her own house). This in turn requires the development of smart grids which optimize distribution, thus minimizing the energy loss due to transportation. This process is quite complex, not only from a modelling point of view but also in terms of enabling authorization, safety, and control processes. Currently the process of energy production and distribution is largely managed and controlled centrally with local energy producers disconnected from the grid and thus potentially wasting any excess of energy they will ever produce. A social computer would allow a more distributed process where production, distribution and consumption are locally controlled in a way to achieve the global societal goal the need of optimizing energy management.

Another example is the intelligent management and control of traffic. Differently from the current situation where the decision is of the single person or of some central authority, the situation could be much more fluid and left to multiple deciders who are aware of local traffic conditions (e.g., a school knows the moment of most intensive traffic in the nearby streets).

Increasing quality and availability of a scarce resource. Proteomics data analysis is underpinned by a small number of curated databases, each maintained as a service in a traditional, centralised style. Data is input to a service; curated by a small team of specialists; then supplied on demand to clients of the service. As the use of proteomics data has increased we reach a limit to the quality of curated data (since the extent of quality control is limited by the size of the central curation team) and to the number of clients that can be served (since this is limited by the size of the server farm serving the data, and the curation task itself is not a lucrative activity because it cannot be easily focused on the very specific needs of clients). A social computation solution is to make available to every proteomics client a system that allows it to acquire proteomics data from other clients (including the original curated databases) provided that a portion of its unwanted but analysed data is made available for acquisition by other peers. This generates value for the group at little cost to each participant because in proteomics only a fraction of the data analysed for a given task is actually of value to the person doing the analysis; the rest is wasted unless shared. With a system for assessing the reputation of each participant, it becomes possible to use the social group for curation of the data rather than relying on centralised curation, which remains but only as one participant in a much larger social network.

Another example falling under this heading is the handling of rare diseases, with the specific knowledge behind in the minds of a very small number of experts whose existence is largely unknown to most doctors.

Occupying untapped economic niches. Clinical trials are a major contributor to the cost of drug development and typically are performed in a centralist style: a workflow is developed by a trial organisation; vetted by specialists; enacted at selected geographical sites which are paid for their participation; then the results analysed. In practice, this approach lacks the agility to cope with diseases, such as malaria, where the geography of outbreaks changes over time and where development of resistance by the malaria parasite often outpaces the speed of clinical trials for new drugs to combat it (since by the time a foolproof trial has been devised, vetted and resourced the parasite may have mutated). A social computation solution is to allow anyone to choose to take part in an open trial but with payment being a function of the effectiveness of the trial, so participants are only well paid if they supply the quality of information that supports the subsequent analysis. This avoids the problem of trials being an "all or nothing" activity (since some data is acquired whether or not it is at a level sufficient to validate a clinical trial) while retaining the ability rigorously to vet and analyse data.

Opportunistic cooperation towards a common goal.[3] Let us consider an hypothetical example where a certain number, possibly quite large, of

[3] Thanks to John Mylopoulos for suggesting this example.

SMEs decides to cooperate as part of a virtual corporation in order to be able to attract large international contracts for home furniture. The companies respectively specialize in furniture design, carpentry, upholstery and furniture wholesaling. In order to survive, the new corporation must set up processes for attracting contracts (sales), designing furniture on the basis of contract specs, manufacturing furniture, doing quality control, ensuring that their contracts are completed on time and in accordance with terms. The problem is dealt with in several steps: (i) resources are identified within the companies, and some resources are recruited from outside to fill gaps in expertise/ skills, (ii) business processes are set up to operationalize the basic objectives of the project, (iii) monitoring mechanisms are set in place for quality control, (iv) governance structures are defined for adapting business objectives, policies and processes as the corporation acquires experience.

7. From Artificial Intelligence to the social computer [4]

In our opinion, we have reached a key point in the development of Computer Science and ICT in general. ICT appears to have reached a peak of success, with impressive forecasts for employment in ICT [5] and with major activities, like the Future Internet Initiative, being started worldwide whose main goal is to root ICT technology deep in the workings of the future society.

But because of this major success, the question which arises naturally is: "*What is Next?*" Where should now ICT dedicate its efforts towards building the foundations of the Future ICT, the ICT which will be needed after the Future Internet Initiative (and other similar initiatives, e.g., Artemisia, in the area of embedded systems) will have achieved all its results. As can be seen, for instance from the recent calls of the European Commission (EC) Future Emerging Technologies (FET) in ICT, there are many options at the various levels of abstraction: at the device level (e.g., by using organic materials), at the theory of

[4] The contents of this section are a synthesis and a discussion in light of the social computer proposal of the argumentation originally provided in [4].

[5] As an example, the labour statistics Bureau of the US has projected that from now to 2018 almost 60% of all the jobs in science and engineering will be in ICT.

computation level (e.g., quantum computation, chemical or bio-computation), at the systems level (e.g., robots, embodied intelligence, complex systems).

Concentrating on the system level, which is our area of expertise, this paper suggests the social computer as one possible such choice. And since a good way to guess the future is to learn from the past, at least in order to avoid making the same mistakes, our analysis starts from an analysis of the work done in Artificial Intelligence (AI), namely the area where, many years ago, the authors began their research careers.

We started with the goal of building some form of AI. In particular, what was called "strong AI", had the goal to build an artificial *human-level intelligence*, Our driving metaphor was human intelligence and behavior, and thus we soon changed the original goal into that of building a *human-like intelligence*. By this we mean that most work in strong AI was (and still is) rooted in the assumption that the first artificial intelligence had to be an *actor* (e.g., an expert system, a robot, a reasoning system) which would live in *environments* (parts of the world) which were not themselves actors, with a clear distinction between what was *inside* or *outside* the artificial intelligence. Furthermore, the science and engineering of strong AI was based on concepts and notions which are metaphors of natural phenomena, such as: goal, plan, action, knowledge, agent, and so on. And the most obvious (only) way to implement these notions was on the existing computers, the most complex and powerful computing machinery available at that time.

The assumptions underlying the social computer are somewhat similar to those underlying strong AI, as briefly described above. However, the original dream of building a human-level intelligence failed. So,why should this not be the case with the social computer? What is the key difference?

The main reason for the failure of strong AI was that the implementation of these human-like notions on a computer run into major problems, most noticeably, time and space scalability. Even the most sophisticated programs would not perform acceptably under the requirements dictated by real world scenarios. In our view, this was no accident. Given the intrinsic combinatorial nature of the world and, consequently, of computation, the implementation of very abstract notions is bound to exhibit a combinatorial behavior unless, by fine tuning between the underlying hardware and software components, one makes sure that the upper level abstractions "fit naturally" the lower level computational structures. On the basis of these considerations, given how little knowledge we still have about the human brain, and given the fact that this situation is not set to change for a while, *we do not see as feasible in the short term a plan which aims at building intelligent artifacts* (e.g., companions, team players) *with general purpose capabilities based on existing computer systems*. They will have to be confined to niche areas and limited specific tasks.

At the same time, these past years have brought us the Internet and the Web. If we look at the Web as analogous to a nervous system which connects its sensing and acting devices into the real world, namely people and sensors, we have now a new form of computing machinery (hardware) of a complexity comparable to that of the human brain. Furthermore (and this is the really good news) differently from the human brain, we understand the plumbing, namely how the people and sensors interact via the nervous system. We have built it! This gives us the unique opportunity to re-visit, with some hope for success, the earlier dream of building a human-level intelligence (which, most likely will turn out not to be a human-like intelligence). The crucial design decision is to make sure that the underlying hardware and the more abstract software notions are isomorphically matched. But this is exactly what our proposal of building the social computer is all about! How far we will go in this path is unclear. At the same time, this project will lay the foundation of the future ICT and, as we discussed in Section 4, we will most likely have a positive impact on our capability to produce technological, service and social innovation.

Acknowledgements

This paper has benefitted from feedback and interactions with many people including. Stuart Anderson, Peter Apers, Roberto Bona, Fabio Casati, Mario Diani, Frank Van Harmelen, Wide Hogenhout, Fabio

Massacci, Daniele Miorandi, John Mylopoulos, Fabrizio Sestini, Paolo Traverso and Enrico Zaninotto.

References

1. Fausto Giunchiglia, "Managing Diversity in Knowledge" (invited talk), European Conference on Artificial Intelligence (ECAI), Trento, September 2006. Lecture Notes in Artificial Intelligence. Online presentation, from http://www.disi.unitn. it/~fausto/knowdive.ppt
2. James Surowiecki, "The Wisdom of Crowds", Anchor Doubleday publisher, 2004.
3. Wikipedia. "Human Flesh Engine". http://en.wikipedia.org/wiki/Human_flesh_search_engine
4. Fausto Giunchiglia, "The future of AI", Knowledge Representation and Reasoning (KR&R) (invited talk), Sidney, September 2008; *and* Symposium for Alan Bundy (invited talk), Edinburgh July 2008. Online presentation from: http://www.disi.unitn.it/~fausto/futureAI.pdf
5. Fausto Giunchiglia, "Trentino as a Lab (TasLab) - Innovation as the way of being, thinking and evolving", Presentation, March 2008, Slides available from the author. Site of TasLab: www.taslab.eu
6. DG Infso, European Commission, "ICT – Information and Communication Technologies, Work Programme 2007-08, Objective ICT-2007-8.6: FET Proactive 6: ICT forever yours", 2006
7. Fausto Giunchiglia (coordinator) "Living Knowledge – Facts, Opinions and bias in time", FP7 FET IP, http:// livingknowledge-project.eu/, duration: 2009-2011
8. Dave Robertson (coordinator) "Open Knowledge", FP6 STREP, http://www. openk.org/, duration: 2006-2008
9. The Amazon Mechanical Turk. https://www.mturk.com/mturk/welcome
10. Amazon Mechanical Turk Monitor. "Jim Gray's search". http://mechanical-turk.blogspot.com/2007/02/you-can-help-find-jim-gray-from-home.html
11. DARPA Network Challenge. https://networkchallenge.darpa.mil/Default.aspx

12. Open Service Oriented Architecture Collaboration http://www.osoa.org
13. "Reality Mining" article in MIT Technology Review special report on emerging technologies 2008 http://www.technologyreview.com/specialreports/specialreport.aspx?id=25
14. P.N Johnson-Laird, "Mental Models", Cambridge University Press, 1983.
15. John Haugeland (editor), "Mind Design", The MIT Press, 1981
16. Marvin Minsky, "The Society of Mind", Simon & Schuster, 1985.
17. Allen Newell and Herbert A. Simon, "Computer Science as empirical enquiry. Symbols and Search", Turing award Lecture 1975, Communications of the ACM, 19 (March 1976), 113-126.