Some results on Multiagent Algorithms in Social Computing/Software Context

Aizhan Satekbayeva[1], Nikolay Shilov[2,3]

[1]L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
satekbayeva@gmail.com
[2]A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia
shilov@iis.nsk.su
[3]Nazarbayev University, Astana, Kazakhstan
nikolay.shilov@nu.edu.kz

Abstract. Distributed system is a group of decentralized interacting executers. Distributed algorithm is the communication protocol for a distributed system that transforms the group into a team to solve some task. Multiagent system is a distributed system that consists of autonomous reactive agents, i.e. executers which internal states can be characterized in socio-human terms Believes (B), Desires (D), and Intentions (I). Multiagent algorithm is a distributed algorithm for a multiagent system.

In this paper we consider two examples of multiagent algorithmic problem in Social Software context. The first example is called Rational Agents at the Marketplace or Cut Cake[1] Problem, the second example is called Mars Robot Puzzle. Some multiagent algorithms for these problems were previously suggested and verified by N. Garanina and N. Shilov in 2011, some results about information exchange were studied by A. Bernstein also in 2011. This time we address a social issue of agent anonymity and privacy in these algorithms and also enforce previous results related to information exchange.

Keywords: multiagent systems and algorithms, assignment problem, safety and progress properties, algorithm verification, privacy, anonymity, Social Software, Social Computing

1 Multiagent systems and algorithms

Many multiagent algorithmic problems can be considered in a Social Computing/Software context, a relatively new research paradigm, the essence of which is as follows. In the modern world many social procedures and processes have algorithmic nature, they look like computing by groups "people" or agents. Therefore, these procedures can be represented in a semiformal pseudocode, their requirements and properties can be (semi)formaly specified. Then the properties of these procedures can be validated by means of formal theories for algorithm/program analysis and verifica-

[1] Sic! "Cut Cake", not "Cake Cutting".

tion. Well, the results of the formal analysis or verification can be interpreted in social terms and may have social significance. And although about multiagent systems and Social Computing/Software started talking in the last two decades, but it is possible to consider as the first example of research in these paradigms the famous Cake-Cutting Problem [4] studied by a group of Polish mathematicians, Hugo Steinhaus, Bronislaw Knaster and Stefan Banach in late 1930s.

Distributed system is a group of decentralized interacting executers. Communication in a distributed system is said to be fair, if any executer that wants to communicate with another one will eventually communicate. Distributed algorithm is the communication protocol implementing a distributed system to solve some task [15]. Multiagent system is a distributed system that consists of agents[2]. An agent is an autonomous rational reactive object (in Object Oriented sense) whose internal state can be characterized in terms of agent's Believes (B), Desires (D), and Intentions (I). Multiagent algorithm is a distributed algorithm that solves some problem by means of cooperative work of agents.

Agent's Beliefs represent its "knowledge" about itself, other agents and an "environment"; this "knowledge" may be incomplete, inconsistent, and (even) incorrect[3]. Agent's Desires represent its long-term aims, obligations and purposes (that may be controversial). Agent's Intensions are used for a short-term planning. Reactivity means that every agent can change its Believes, Desires, and Intentions after communication with other agents and environment, but every agent is autonomous, i.e. any change of its Believes, Desires and Intensions depends on the agent itself, and a change can't be decreed from outside. Agents of the described kind are called BDI-agents [17].

A rational agent has clear "preferences" and always chooses the action (in feasible actions) that leads to the "best" outcome for itself; a bounded rationality is "decision making" limited by the cognitive and deductive abilities of agents or other constrains (e.g. amount of time they have to make decisions).

An intelligent agent [16] is the agent, with not an imperative deterministic program, but declarative non-deterministic logic program consisting of clauses in the form $C_1 \& \dots C_k \Longrightarrow$ fork $A_{1,1}; \dots A_{1,m} || \dots A_{n,1}; \dots A_{n,m}$ join, where $C_1, \dots, C_k$ - logical conditions on the local variables and communication channels of the agent, fork ... || ... ... join - construction for parallel branching, $A_{1,1}; \dots A_{1,m}, \dots A_{n,1}, \dots, A_{n,m}$ - actions like assignments to local variables, reading/writing data from/to input/output channels.

2 Case Studies: agents at the marketplace and on Mars

First let us discuss in brief the following problem that is called in [13] Rational Agents at the Marketplace (RAM-problem).

[2] i.e. executers are agents

[3] According to Plato, knowledge true belief, i.e. a judgment that has a validation.

There are $n > 1$ customers (buyers) and the same number of goods at the marketplace, each good has individual price for different buyers. Every buyer $b \in [1..n]$ is a rational agent, who wants to buy exactly one good, he knows in advance his price-list $\{p(b,g) \geq 0 : g \in [1..n]\}$ (as a private information), but is not aware about any price offer to any other customer. All buyers are rational agents that can communicate with other buyers in P2P-manner, negotiate and flip (change individual purchase intensions) so that all flips must always be rational for participating agents. Since all buyers are rational agents, each buyer $b \in [1..n]$ confesses that time is money and has an individual "fine" $f(b)$ for every unproductive negotiation with any other buyer. However, each buyer can make a deal (or purchase) if and only if he (the buyer) knows that no other customers will ever attempt to buy this good.

***Problem**: Design a multiagent algorithm, which allows each buyer sooner or later to purchase some unit of good.*

RAM-problem is illustrated on the Fig.1 below. Here we have 3 buyers (depicted to the right) and three goods (depicted to the left), each buyer knows his own individual price-list, but also has his current intension to buy a particular good. (Current intensions are represented by red arrows.) Several buyers may have intension to by the same good. Conflict of intensions is depicted by lightning on the Fig. 1. RAM-problem asks us to resolve conflicts of this kind.

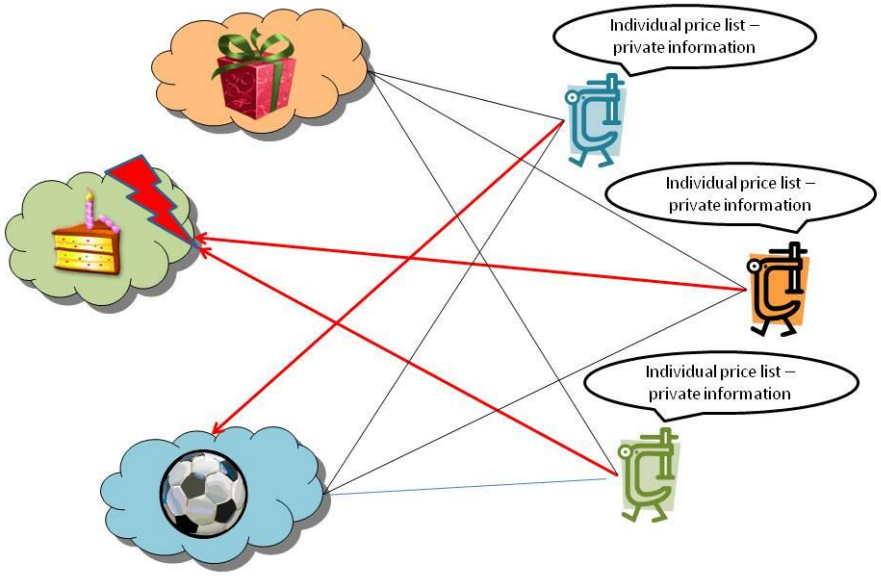

Fig. 1 – Rational agents on the Marketplace

RAM was solved in [13] for the agents presented by imperative programs. It "grew up" from the following problem considered in [3][12] and called *Mars Robot Puzzle* (MRP).

> *There are* $n > 1$ *autonomous agents ("robots") and the same number of shelters in general position*[4] *on a plain part of Mars. Locations of all shelters are fixed and known to all robots. Each robot knows a shelter that was assigned to it from the very beginning*[5]*, its own distances to all shelters, know all other robots, but doesn't know locations of any other robot. Robot can communicate with each other in P2P-manner and every pair of communicating robots may swap their shelters. All robots have to select individual shelters to move in by a straight route*[6]*. Definitely, robots should not collide (it means that their routes should not intersect). Hence, every individual robot can move to a shelter only when it knows for sure that it will not collide with any other robot on the route.*
>
> ***Problem****: Design a multiagent algorithm that guarantees that every robot will eventually know that its route to the selected shelter does not intersect with routes of other robots.*

MRP is illustrated on the Fig.2 below. Here we have 3 robots (enumerated by numbers) and three shelters (indexed by letters), each robot knows his own individual location, but also has his current intension to go into a particular shelter (depicted by a red arrow). It is assumed that from the very beginning all robots have individually assigned shelters, and that robots can swap shelters in pairs; it implies that all robots always have disjoint intensions (current shelters), but routs to these shelters may intersect. Conflicts of intensions are depicted by lightning on the Fig. 2. MRP is to resolve conflicts of this kind. One can observe that RAM-problem can be considered as a "preprocessor" for MRP: MRP assumes that robots have individual shelters assignments, but RAM is about how agents can make an assignment of this type.

MRP multiagent interpretation of the Dijkstra problem [14]: *Design a algorithm that connects by straight segments n black ("robots") and n white ("shelters") points on a plain so that segments don't intersect.* But MRP can be considered also as a special geometric interpretation the assignment problem in Graph Theory, or to the convex hull problem in Combinatorial Geometry. At the same time MRP is closely related to the path-planning problem [8].

---

[4] i.e. none three objects lie on a straight line

[5] The initial assignment is individual, i.e. a one-to-one assignment.

[6] Assume that there are no obstacles like rocks, holes, etc. between any robot and any shelter.

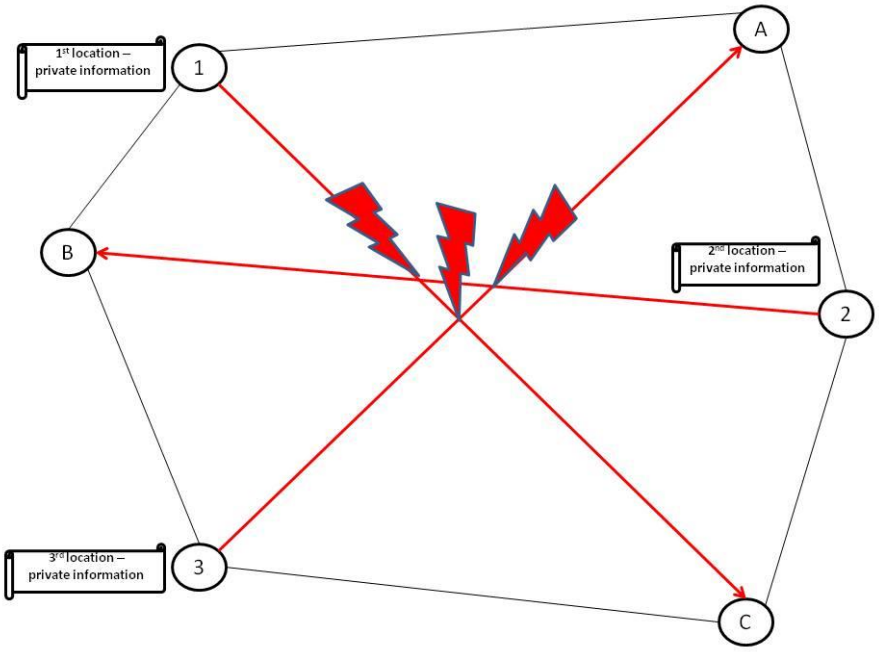

Fig. 2 – Mars Robot Puzzle

3 How to solve RAM and MRP

The difference between RAM and MRP is manifold. First, agent rationality in RAM can be represented numerically by their price-lists, while in MRP agents do not care about numbers at all, but about safety on their routs to shelters. Next, MRP has a clear geometric interpretation, but it is not clear from the very beginning, whether any intersection-free set (of routes) exists, and, hence it is not obvious that the problem has a solution. In contrast, RAM has no a geometric interpretation, but it is seems a priori that some protocol may exist.

Nevertheless, from the algorithmic point of view, RAM and MRP are closely related, since their solutions belong to the class of so-called wave algorithms [15]. This class of distributed algorithms is characterized by the following general properties.

- Termination: all agents complete their work within a finite time.
- Decision: each agent has the final decision.
- Dependence: the decision of each agent influences all agents.

Papers [3][12] presented and proved imperative wave algorithms SWP (SWaPping) for Mars Robot Puzzle, and paper [13] presented proved imperative wave algorithms LSM (Look for Salesman) for Rational Agents at Marketplace. Individual beliefs of every agent in both algorithms are represented by two integer counters NC and CF:

- current value of NC (Number of Conflicts) is an upper estimation of the number of agents with whom the agent may have a conflict of intensions right now;
- current value of CF (Conflict Free) is a lower estimation of the number of agents that have no conflicts at all.

An agent believes that it doesn't have conflicts with anybody when NC=0; it believes that there is no any conflict in the system when NC=0 and CF=$2 \times (n-1)$ (i.e. it checks twice that all other agent also believe that they don't have conflicts.)

Agent part in both algorithms LSM and SWP follows:

```
DO CF:=0;
DO NC:= (n − 1);
contact all other (n − 1) agents in turns and with each
partner
DO IF no conflict with a partner THEN NC:=NC-1
       ELSE resolve the conflict and NC:= (n − 1)
UNTIL NC=0;
//i.e. the agent believes that it is conflict-free
contact all other (n − 1) agents in turns and with each
partner
DO IF no conflict with a partner and the partner believes
it is conflict-free
       THEN CF:=CF+1
       ELSE resolve the conflict, NC:= (n − 1) and CF:=0;
DO IF no conflict with a partner and the partner believes
it is conflict-free
       THEN CF:=CF+1
       ELSE resolve the conflict, NC:= (n − 1) and CF:=0
UNTIL CF=2 × (n − 1)
//i.e. the agent believes that system hasn't conflicts.
```

System part in both algorithms LSM and SWP is to provide *fair* communication, i.e. any time when an agent would like to communicate with another agent they eventually will communicate.

LSM algorithm uses the following game-theoretic approach [1] for conflict resolution between two agents. Let x and y be two agents that have intension to buy one and the same good g. Each agent z in $\{x, y\}$ has two strategies in negotiations – to bid_z for the good g or to $flip_z$ to the next good in the private price-list. Let f_x and f_y be individual fines of x and y for non-productive negotiations (when both either bid or flip simultaneously), and let d_x and d_y be their individual losses (if they flip to the next goods in their price-lists). Remark, that f_x and f_y as well as d_x and d_y are non-positive. Then let us consider the following strategic game:

$$\begin{pmatrix} x \backslash y & bid_y & flip_y \\ bid_x & (f_x, f_y) & (0, d_y) \\ flip_x & (d_x, 0) & (f_x, f_y) \end{pmatrix}$$

This game has no Nash equilibrium in pure strategies, but we may solve the game mixed strategies. Let agent x play strategy bid_x with probability p and strategy $flip_x$

with probability $(1-p)$, and agent y play strategy bid_y with probability q and strategy $flip_y$ with probability $(1-q)$. Then we have two equations [1]:

$$\begin{cases} f_x \cdot q + 0 \cdot (1-q) = d_x \cdot q + f_x \cdot (1-q), \\ f_y \cdot p + 0 \cdot (1-p) = d_y \cdot p + f_y \cdot (1-p). \end{cases}$$

It implies that

$$\begin{cases} q = {f_x}/{(2f_x - d_x)}, \\ p = {f_y}/{(2f_y - d_y)}. \end{cases}$$

Recall that $f_x, f_y, d_x, d_y \leq 0$, $0 \leq p, q \leq 1$. Hence

$$\begin{cases} 0 \neq 2f_x - d_x \leq f_x, \\ 0 \neq 2f_y - d_y \leq f_y, \end{cases}$$

or $0 \neq f_x \leq d_x \leq 0$ and $0 \neq f_y \leq d_y \leq 0$. In other words: the absolute value of agent's fine can't be 0 and less then the absolute value of its loss when it flips to the next good in the price-list.

The following proposition about LSM is proved in [13].

Proposition 1.

If *a multiagent system with fair communication consists of $n > 0$ buyers each of which would like to buy a single of n goods, it is common knowledge (in the system) that all buyers are agents executing algorithm LSM, and for every buyer b its fine $f_b \neq 0$ and its absolute value isn't less than the maximal absolute value of itsloss when it flips to the next good in the price-list,*

then *every agent will eventually terminate, and upon termination its beliefs that it hasn't conflicts and the system is conflict-free are true*.

SWP algorithm resolves agent conflicts much easier. Let x and y be two robots, s and t be their intentional shelters. Then let x and y to swap shelters s and t, if $|(x,t)| + |y,s| < |x,s| + |(y,t)|$. The following proposition is proved in [3][12].

Proposition 2.

If *a multiagent system with fair communication consists of $n > 0$ robots each of which knows some initial individual shelter, it is common knowledge (in the system) that all robots are agents executing algorithm SWP and all robots and shelters are in general position,*

then *every robot will eventually terminate, and upon termination its beliefs that its straight rout to selected shelter is safe*[7] *and the system is conflict-free are true*.

These two propositions solve RAM and MRP problems, but under assumption of fair communication. This assumption constitutes that if anyone of agents ever wants to communicate with any another agent, sooner or later the communication session between them will surely take place. In this paper we won't discuss how to guarantee

[7] i.e. hasn't intersections with routs of other robots

this fairness, but we would like to point to one option that solves the problem: one can assign priorities to agents and allow seniors to initiate communication with juniors.

Another drawback of the algorithms LSM and SWP is their imperative nature. And, therefore, agents, guided by these algorithms are rational, but have not yet become intelligent.

4 Social Software and Agents

At the same time RAM is closely related to the classic Cake Cutting Problem (CC-problem), also known as Fair Division Problem [4] that has been introduced by a group of Polish mathematicians, Hugo Steinhaus, Bronislaw Knaster and Stefan Banach.

The CC-problem is to divide an infinitely dividable resource ("cake") in such a way that all recipients believe that they have received a fair. Special cases of the problem are proportional and envy-free division. A division is said to be envy-free if each recipient believes (at the moment of reception) that according to his measure no other recipient has received more than he has of a heterogeneous cake; in contrast, a proportional division deals with a homogeneous cake where each of m recipients have to receive exactly $1/\mathrm{m}$ of the cake's volume.

To explain connections between RAM- and CC-problems, it is enough to reformulate a RAM as follows:

> *The cake is cut on $n > 0$ pieces, which should be divided among $n > 0$ recipients. Each recipient in $b \in [1..n]$ is the intelligent rational agent to whom exactly one of piece of a cake is necessary, and it knows the scale of value of pieces $\{p(b,s) \geq 0 : s \in [1..n]\}$. All recipientscan (in P2P-manner) communicate, negotiate, make concessions, flip (individual change) their pieces of a cake so that all flips must be rational. However, each recipient can buy the chosen piece if and only if he/she knows that nobody else will ever apply for this piece of cake.*
> ***Problem**: Design a multiagent algorithm for recipients, which will allow each agent sooner or later to get a piece of cake.*

Differences between RAM and CC problems are evident: in CC-problem a cake is an infinitely dividable resource, while in RAM-problem a "resource" has been cut already onto "salesmen"; solutions of the CC-problem may be sequential, while solutions (if any) of RAM must be multiagent (i.e. distributed, parallel and concurrent) by the problem statement. But even multiagent solution of CC-problem can be unsuitable for a RAM-problem. For example, the classical envy-free solution of CC-problem for two participants consists in the following: one agent cuts the cake so that any of two pieces will satisfy him/her, and the second chooses from these two pieces which satisfies her/him. As it is easy to see if the cake is already cut on two pieces, and both agents wish the same piece, the system of these two agents will get to the deadlock.

But in spite of these differences, RAM and CC problems have something in common since they both are examples of a new research paradigm of Social Software [9].

In the modern world very many social requirements and procedures have algorithmic character. These requirements can be written as (semi-)formal specifications and procedures – software (in a pseudocode). Then the properties of these procedures can be analyzed and verified by formal methods. Well, the results of the formal analysis or verification may be interpreted in socially significant terms. And though about Social Software started talking only in a XXI century, but it is possible to consider as the first example of application of this paradigm research of the Cake Cutting Problem by H. Shteinhaus, B. Knaster and S. Banach.

5 Privacy in RAM-problem

Let us assume that the number n of agents and all distances are integer numbers presented by fixed finite number of digits in any (fixed) position notation. Note that in this case the meaning of SWP protocol consists in computation of n integer functions of individual price lists. By the theorem of Oblivious Transfer [5],[6],[10],[17], there is a way to compute this function, in which the participants will not send full information about their distances to each other. Unfortunately, a direct description of this method is a cumbersome. Fortunately we can construct a protocol of swapping, in which the participants will not disclose to each other information about distances thanks to our assumption about fixed size of data representation (i.e. number of agents and individual prices): due to this assumption all functions to be computed are Boolean functions of Boolean arguments. The construct presented below is some modification of construction presented in [2].

Assume we have to compute a Boolean function f such that some part of its arguments is known to the agent A, and the other part – to the agent B. Thus participants don't want to open each other information on their individual data. Assume that function f is written using conjunctions &, negations $\neg$ and exclusive disjunction $\oplus$. Let us represent the function f by logic scheme where inputs are arguments, and the output is the function value. We will calculate the values in the nodes of the scheme, but not completely: each node i will receive the bits a_i and b_i from agents A and B such that the value of a node is equal to $a_i \oplus b_i$, with bit a_i known only to A and bit b_i only B. For this purpose agents proceed as follows.

For each input bit of p each agent generates two random bits a and b such that $a \oplus b = p$, and sends one of these bits to his partner. Thus, the input data are divided.

Now it was necessary to describe how to compute the results of Boolean operations. Situation with negation and exclusive disjunction is simple: to compute result of negation, it is necessary to negate the corresponding bit of the agent; to compute result of exclusive disjunction each agent has to compute exclusive disjunction for his/her part.

Conjunction is more complicated. Let on the input there are bits of a_1, b_1, a_2 and b_2, and it is required to receive such bits a and b that $a \oplus b = (a_1 \oplus b_1) \& (a_2 \oplus b_2)$. As excluding or and conjunction is an addition and multiplication on the modulo 2 re-

spectively, we have $(a+b) = (a_1 \times a_2 + a_1 \times b_2 + b_1 \times a_2 + b_1 \times b_2) \mod (2)$. As $a_1 \times a_2$ is known to agent A, and $b_1 \times b_2$ - the agent B, they have to compute bits c and d such that $(c+d) = (a_1 \times b_2 + b_1 \times a_2) \mod (2)$; then it is possible to put $a \coloneqq (a_1 \times a_2 + c) \mod (2)$ and $b \coloneqq (b_1 \times b_2 + d) \mod (2)$.

Bits c and d may be computed using the following protocol. Agent A generates a random bit x. Then he believes $s_{00} \coloneqq x$, $s_{01} \coloneqq (x + a_1) \mod (2)$, $s_{10} \coloneqq (x + a_2) \mod (2)$, $s_{01} \coloneqq (x + a_1 + a_2) \mod (2)$. With data transfer protocol "blindly 1 of 4" (oblivious transfer) [5],[6],[10],[17] agent B knows value of s_{b_1,b_2}. Now it is possible to put $c \coloneqq x$ and $d \coloneqq s_{b_1,b_2}$.

The above arguments prove the following proposition.

Proposition 3. *Let us assume that the number n of agents and all distances are integer numbers presented by fixed finite number of digits in any (fixed) position notation. There exists a variant of SWP protocol in which the agents do not disclose to each other their individual distances to shelters.*

6 Conclusion

Above (a footnote 3) we have already referred to Plato's authority, having defined knowledge as a true belief about the reality. However, it is not the only interpretation of the concepts "belief" and "knowledge" of the agent.

Apparently, the first to propose a formalization of these concepts was a Finnish logician J. Hintikka [7]. In his interpretation the definition of these concepts is to be associated with each agent indistinguishable binary relation between the possible states of the environment, "belief" corresponds to the symmetric and transitive relations and "knowledge" - reflexive, symmetric and transitive relations. More from this approach can be found on the monograph [5].

Essential lack of Hintikka's approach to formalization of concepts of knowledge of belief is high complexity of the verification (by model checking). Unfortunately, verification of multiagent systems with the use of their opinions (belief) and knowledge has non-elementary (lower and upper) complexity [11] while the verification of multiagent systems without the use of belief and knowledge has an elementary complexity (see for example [16]).

Acknowledgement. This research is supported by gran numb. 13-01-00643-a of Russian Basic Research Foundation.

References

1. Apt K., Gradel E. (Ed.) Lectures in Game theory for Computer Scientists. Cambridge University Press (2011)
2. Bernstein A.Yu.: Informatsyonny i kriptograficheskiy aspekty zadachi o robotah na Marse. In: Ershovskaya konferentsiya po informatike: Trudy tretego seminara Znaniya i Ontologiya *ELSEWHERE*, pp. 35-46. Prais-Kurer, Novosibirsk (2011)

3. Bodin E.V., Garanina N.O., Shilov N.V.: Mars Robot Puzzle (a Multiagent Approach to the Dijkstra Problem). Modeling and analysis of information systems 18(2), 111-126 (2011)
4. Brams S.J. and Taylor A.D.: Fair Division – From cake-cutting to dispute resolution. Cambridge University Press (1996)
5. Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.: Reasoning about Knowledge. MIT Press, London (1995)
6. Goldreich O.: Foundations of Cryptography - A Primer. Foundations and Trends in Theoretical Computer Science 1(1), pp. 1–116 (2005)
7. Hintikka J.: Knowledge and Belief. Cornell University Press (1962)
8. LaValle S.M.: Planning Algorithms. Cambridge University Press (2006)
9. Parikh R.: Social Software. Synthese, 132, pp.187-211 (2002)
10. Schneier B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Willey&Sons, New York (1996)
11. Shilov N.V., Garanina N.O.: Modal Logics for reasoning about Multiagent Systems. In: Encyclopedia of Artificial Intelligence. J.R. Rabucal, J. Dorado, A.P. Sierra, editors. Information Science Reference, pp.1089-1094 (2008)
12. Shilov N., Garanina N., Bodin E.: Multiagent approach to a Dijkstra problem. In: CS&P'2010 Workshop on Concurrency, Specification and Programming, No.237, vol.1, pp. 73-84. Humboldt-Universitat zu Berlin. Informatik-Bericht, Helenenau (2010)
13. Shilov N., Garanina N.: Rational Agents at the Marketplace. Has been accepted for presentation at Workshop on Concurrency, Specification and Programming CS&P'2011, pp.465-476. Bialystok University of Technology,Pułtusk (2011)
14. Shilov N.V., Shilova S.O.: Etude on theme of Dijkstra. In: ACM SIGACT News, vol.35(3), pp. 102-108 (2004)
15. Tel G.: Introduction to Distributed Algorithms. Cambridge University Press, 2nd Edition (2000)
16. Valiev M.K., Dekhtyar M.I., and Dikovsky A.Ya.: Systems of Agents Controlled by Logical Programs: Complexity of Verification. Programming and Computer Software, 35(5), pp.266-281 (2009)
17. Wooldridge M.: An Introduction to Multiagent Systems. John Willey&Sons Ltd, New York (2002)