Rational Agents at the Marketplace* (extended abstract)

N.V. Shilov and N.O. Garanina

A.P. Ershov Institute of Informatics Systems,
Lavren'ev av., 6, Novosibirsk 630090, Russia,
{garanina, shilov}@iis.nsk.su

Abstract. We study multiagent algorithms for the following problem: There are buyers (customers) and salesmen at the marketplace; all customers are rational agents; in contrast, all salesmen are not agents, but every salesman has individual price for every buyer; the problem is to define a rationality-based protocol of pairwise negotiation, flips/swaps (of salesmen) between agents that leads every agent (1) to knowledge about its individual salesman, and (2) to knowledge that it is impossible to reduce the total price to be paid by swapping salesmen. We call this problem *Rational Agents at the Marketplace*. The problem is related to the classical *Cake-Cutting Problem* (also known as *Fair Division Problem*), and to *Mars Robot Puzzle* that we examined previously.

1 Introduction

A *multiagent system* is a distributed system [11] that consists of *agents*. An agent is an autonomous reactive object (in OO-sense) whose internal states could be characterized in terms of *Beliefs* (B), *Desires* (D), and *Intentions* (I). Agent's Beliefs represent its "knowledge" about itself, other agents and an "environment"; this "knowledge" may be incomplete, inconsistent, and (even) incorrect. 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 could change its Beliefs, Desires, and Intentions after communication and interaction with other agents (including an environment), but every agent is autonomous, i.e. a change of "personal" Beliefs, Desires and Intentions is not decreed by other agents. Agents of the described kind are usually called BDI-agents [13]. 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 abilities of agents (e.g. the finite amount of time they have to make decisions) [7]. A *multiagent algorithm* is a distributed algorithm [11] that solves some problem by means of cooperative work of agents in a multiagent system.

* The research has been supported by Russian Foundation for Basic Research (grant 10-01-00532-a) and by Siberia Branch of Russian Academy of Science (Integration Grant n.2/12).

CONCURRENCY, SPECIFICATION AND PROGRAMMING
M. Szczuka et al. (eds.): Proceedings of the international workshop CS&P 2011
September 28-30, Pułtusk, Poland, pp. 465-476

In this paper we exam the following problem that we call *Rational Agents at the Marketplace* (RAM-problem or RAMP).

> There are $m \geq 1$ customers (buyers) and $n \geq 1$ salesmen (traders) at the marketplace. Every salesman has a (single) unit of some indivisible good for sale, and every buyer wants to buy exactly one unit of the good. All salesmen offer their units of the good individually to buyers by individual prices $\{p_{s,b} \in \mathcal{N} : 1 \leq b \leq m$ is a buyer, $1 \leq s \leq n$ is a salesman$\}$. At the very beginning every buyer chooses its individual salesmen. All buyers are rational agents that can communicate, negotiate, make concessions, and flip (change) and swap their individual salesmen pairwise (and only pairwise) in peer-to-peer (P2P) manner so that all concessions and swaps must be rational for both participating agents.
> Problem: Design a multiagent algorithm (protocol) for negotiations and concessions, changes ("flips") of salesmen, exchanges ("swaps") of salesmen by buyer that eventually leads every buyer to knowledge about its individual salesman with whom it could make a beneficial (i.e. rational) deal
> 1. without conflicts (concurrency) with other buyers, and
> 2. no pairwise swap can improve (reduce) the total price payed by buyers.

This problem grew up from our study of the following *Mars Robot Puzzle* (MRP) [9].

> There are $n > 1$ autonomous agents ("robots") and (the same) number of shelters on a plane part of Mars. Locations of all shelters are fixed and known to all robots. Every robot could communicate with any other robot in P2P manner. Every robot knows its own position, but is not aware about positions of other robots.
> At some moment all robots fix their current positions, and have to select individual shelters to move at by a straight route. Assume that there are no obstacles (like rocks, holes, robots and shelters, etc.) between any robot and any shelter. Definitely, robots should not collide (it means that their routes should not intersect). Hence, every individual robot can move to its 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 (and hence robots will not collide in a motion).

The difference between RAMP and MRP is manyfold. First, in RAMP agents are assumed to be rational, while in MRP agents do not care about their benefits (preferences) at all. 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 a desired protocol may exist. Fortunately,

the existence of these routes could be proved by contradiction, or by reduction to the *assignment problem* in Graph Theory [3], or to the convex hull problem in Combinatorial Geometry. At the same time MRP is closely related to the path-planning problem in Artificial Intelligence. In contrast, RAMP has no a geometric interpretation, but it is seems a priori that some protocol may exist.

Nevertheless, RAMP and MRP are closely related due to the core protocol SMEx for MRP [9]. This protocol is a multiagent variant of a local search algorithm for a combinatorial geometry problem suggested by E.W. Dijkstra [10]. We presented in the paper [9] a series of modifications of SMEx protocol that solve MRP problem, and proved (manually) their correctness. All these modifications belong to a class of so-called *wave algorithms* [11], since they meet the following properties:

- a termination: each computation is finite;
- a decision: each computation contains at least one decide event;
- a dependence: each event potentially influences all computations.

All our algorithms in this paper rely upon the following fairness communication assumption [9]: communication (in a multiagent system) is said to be *fair*, if every agent which would like to communicate with any other agent will communicate eventually[1]. We distinguish *belief* and *knowledge* notions according to the famous Plato thesis: *Knowledge is true belief*. Thus our approach to *knowledge* and *belief* is not very formal like in [5, 4], but nevertheless we *believe* that it could be formalized in terms of Kripke structures and indistinguishable worlds. We also assume, that the total number of customers m and the complete set of customers is *common knowledge* in this group of agents (customers) [4], and that every customer $b \in [1..m]$ knows its individual price-list (i.e. the set $P_b = \{p_{s,b} \in \mathcal{N} : 1 \leq s \leq n\}$) sorted in the ascending order (i.e. the cheapest first, the most expensive – the last). We interpret this price-list as buyer's preferences, its rationale for flipping/swapping traders. At the same time we assume a bounded rationality as follows: a buyer never mind to remember data of other customers and always is looking for the most rational *local* action (i.e. just one step ahead). We use pseudocode a-lá [11] to present algorithms (protocols) in this paper.

At the same time RAMP is closely related to the classic *Cake Cutting Problem* (CC-problem, also known as *Fair Division Problem*) that has been introduced by a group of Polish mathematicians, Hugo Steinhaus, Bronislaw Knaster and Stefan Banach [2]. 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 amount. A special cases of the problem are *proportional* and *envy-free* division. A division is said to be envy-free if each recipient believes 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 $1/m$ of the cake's volume.

[1] In terms of Linear Temporal Logic: **F**((agent 1 wants to talk to agent 2) **U** (agent 1 and agent 2 are talking))

Differences between RAMP and CC-problem are evident: in CC-problem a cake is an infinitely dividable resource, while in RAMP-problem a "resource" has been cut already onto "salesmen"; solutions of the CC-problem may be sequential, while solutions (if any) of RAMP must be multiagent (i.e. distributed, parallel and concurrent) by the problem statement.

But in spite of these differences, RAMP and CC-problem have something in common since they both are examples of a new research paradigm of *social software* [6]. This interdisciplinary research paradigm borrows tools and techniques from game theory, computer science, theoretical programming for formal analyzes and design of social procedures. We believe that our paper will contribute to this field of research by study of the RAMP-problem.

The rest of the paper is organized as follows. The next section 2 sketches a protocol for "initial" choices of salesmen by buyers (i.e. that satisfies the first constraint of RAMP). The following section 3 presents a protocol for the most "rational" for buyers distribution of salesmen that is possible to arrange by pairwise swaps (i.e. that satisfies the second constraint of RAMP). Finally, we conclude in the last section 4 with a discussion of further research topics.

2 Looking for a salesman

In this section we design and prove a multiagent algorithm (protocol) for rational negotiations between buyers that eventually leads every buyer to knowledge about its individual salesman with whom it could make a deal without conflicts (concurrency for a salesman) with other buyers. We assume that all customers (buyers) arrive to the marketplace *simultaneously*. One may consider an alternative opportunity: buyers arrive one by one or (more generally) group by group (one by one or by a single group in particular). But this "more general" situation could be reduced to the simultaneous arrival: since m (the total number of customers) is common knowledge, all buyers may just wait until all have arrived to the marketplace.

The protocol consists of the algorithm LSM (*Look for a SalesMan*) for an individual agents and a communication scheduler. At every moment every buyer has some salesman (current salesman) as its current intention; at the very beginning this intention is the salesman with the best (cheapest) price for this customer. Beliefs of every buyer are represented by two integer counters: NC for *Number of Conflicts* and CF for *Conflict-Free* buyers; NC represents agent's upper estimation of number of buyers with whom it could have conflicts, and, respectively, CF represents its lower estimation of number of agents that have no conflicts at all; in particular, the agent believes that it does not compete for the current salesman with any other buyer as soon as $NC = 0$; the agent believes that there are competition for any salesman at the marketplace as soon as $NC = 0$ and $CF = 2 \times (m - 1)$, i.e. it believes that it has no rivals, and it checks twice that all buyers believe that they do not compete for their salesmen. But in the case when two agents (say i and j) compete for a salesman, then they play the following static game with complete information [1] that we denote as

Flip_Game(i, j):

$i \setminus j$	*bid*	*flip*
bid	(f_i, f_j)	$(0, l_j)$
flip	$(l_i, 0)$	(f_i, f_j)

Two moves (strategies) in this game are to *bid* for the same salesman and to *flip* to the next salesman. Before the game every participating agent $k \in \{i, j\}$ sends to the partner two integer values: l_k is its loss[2], if it flips, and f_k is its fine[3] for simultaneous bidding or flipping with the partner. These individual payoff values l_k and f_k are known to every agent $k \in [1..m]$ a priori. Then both agents compute a mixed strategy Nash equilibrium (that is a rational behavior) and play the corresponding mixed strategies [1] until they make different moves (i.e. one flips, another bids).

Description of a fair scheduler for planning contacts of agents is out of scope of our paper. Pseudocode of the LSM-algorithm follows, but first we would like to comment a meaning of some variables: `Me` is a personal agent's number; `cur_sman, next_sman, par_sman` – the current, next, partner's salesman; `par_bel` – partner's belief that it (the partner) is conflict-free; `contacts` – a set of buyers; `my_loss` – a loss due to flipping salesmen; `par_loss` – partner's loss due to flipping salesmen; `my_fine` – a personal fine for simultaneous bidding/flipping (it is known to the agent); `par_fine` – partner's fine for simultaneous bidding/flipping; `pro_bid` solves the *Flip_Game* in mixed strategies.

```
algorithm LSM::
const Me : integer in [1..m]
const my_fine : integer;
var my_loss, par_loss, par_fine : integer;
var NC : integer in [1..m];
var CF : integer in [1..2*m];
var contacts : set of [1..m];
var partner : integer in [1..m];
var cur_sman, next_sman, par_sman : integer in [1..n];
var par_bel : boolean;
var pro_bid : real in [0..1];
begin
1:  NC:= (m - 1); CF:= 0;
2:  cur_sman := the salesman with the cheapest price in P_Me;
3:  repeat
4:   if NC > 0 then NC:= (m - 1);
5:   contacts:= set of all buyers but Me;
6:   repeat
7:    partner := a buyer in the contacts ready to communicate;[4]
8:    start communication session with the partner:
```

[2] This value must be negative, since the agent flip a better salesman to a worse salesman.

[3] This value must be negative, since *time is money*.

[4] A scheduler resolves this request.

```
9:    { send <cur_sman>, <NC=0> to partner ||
        receive <par_sman>, <par_bel> from partner }
10:   if cur_sman = par_sman then
11:   begin
12:    next_sman:= a salesman with the next to cur_sman's price
              according to P_Me, if it exists, cur_sman otherwise;
13:    my_loss:= P_{cur_sman,Me} − P_{next_sman,Me};
14:    { send <my_loss>, <my_fine> to partner ||
         receive <par_loss>, <par_fine> from partner }
15:    pro_bid := probability of My bid in Nash equilibria
                                          in Flip_Game(Me,partner);
16:    repeat                                    // Play Flip_Game;
17:     my_move:= some_in {cur_sman:pro_bid,
                                        next_sman: (1 - pro_bid)};[5]
18:     { send <my_move> to partner ||
         receive <par_move> from partner }
19:    until (my_move=cur_sman ∧ par_sman≠par_move) ∨
                            (my_move≠cur_sman ∧ par_sman=par_move);
20:    cur_sman:= my_move; NC:= (m - 1); CF:= 0
21:   end
22:   else if NC > 0
23:       then {NC:= (NC - 1); CF:= 0}
24:       else if par_bel
25:           then CF:= CF + 1
26:           else {NC:= (m - 1); CF:= 0}
27:   close communication session with partner;
28:  contacts:= remove partner from contacts;
29:  until contacts becomes empty
30: until (NC = 0 ∧ CF = 2*(m - 1))
end.
```

Some important properties of this LSM-algorithm are accumulated in the following Lemmas 1, 2 and 3. Their proofs will be published in a forthcoming full research report.

Lemma 1. *Assume that two agents i and j play Flip_Game(i,j) where all payoff values l_i, f_i, l_j and f_j are negative*[6]*. Then Nash equilibrium in mixed strategies in Flip_Game(i,j) exists (and is unique) iff $f_1 < l_1$ and $f_2 < l_2$.*

Let remark that lines 13 – 19 of the above algorithm LSM contains a game-theoretic procedure for conflict resolution between the agent and its partner. Let LSM′ be a variant of the algorithm LSM, with any procedure (non-deterministic, probabalistic or deterministic) in these lines for conflict resolution such that

[5] Select My next move according to assigned probabilities.

[6] This assumption is very natural, since losses l_i and l_j are due to choice of the next price in the ascending order, and fines f_i and f_j are due to simultaneous bidding/flipping, i.e. due to loss of time (but *time is money*).

either the agent flips to its next salesman or (alternatively) the partner flips to its next salesman.

Lemma 2. *Assume that it is common knowledge in a system that all agents execute one and the same algorithm LSM′. Then every agent in this system at every moment of time after execution of line 2 always knows that all salesmen that gives smaller price to the agent than its current salesman are intensions of some other agents.*

Let us recall that communication is said to be *fair* in a multiagent system, if every agent which would like to communicate with any other agent will communicate eventually.

Lemma 3. *Assume that all agents of a system execute one and the same algorithm LSM′, and that communication is fair in the system. Then the system eventually terminates.*

The desire of every buyer is to select in a rational way an individual salesman and to know that it could make a deal with the salesman without a competition with other buyers. Lemmas 1, 2 and 3 together imply the following proposition.

Proposition 1.

If *a system with fair communication consists of $m > 0$ buyers each of which would like to make an individual deal with some of $n \geq m$ salesmen, it is common knowledge (in the system) that all buyers are agents that execute algorithm LSM, and for every buyer b its payoff value* `my_fine` *is less than* $\min\{p_{s',b} - p_{s'',b} : s', s'' \in [1..n]$ *and s'' has the next price to the price of s' in $P_b\}$,*

then *every agent will eventually terminate, it will know upon termination that nobody in the system will never compete for its current salesman* `cur_sman`*, and hence it will be able to make a deal with this salesman.*

Proof of this proposition will be published in a forthcoming full research report.

Unfortunately, we do not know yet whether the outcome of the LSM-algorithm is a Pareto-optimal assignment or not. This topic requires more research.

3 Exchange by salesmen

In this section we design and prove a multiagent algorithm (protocol) for rational negotiations between buyers (who already have their individual salesmen) that eventually leads every buyer to knowledge that it is impossible to improve (reduce) by pairwise swapping salesmen the total price all buyers have to pay. Again the protocol comprises two algorithms: SWP (*SWaPping*) for an individual agents and a fair communication scheduler; description of a scheduler is out of scope of our paper, SWP-algorithm is similar to the above LSM-algorithm, it is discussed and presented in the sequel.

At every moment every buyer knows an individual salesman (a current salesman) as its current intention (and it knows that nobody competes for this salesman); at the very beginning this current salesman is known *initial* salesman (for example due to the previous LSM-algorithm). Again, beliefs of every buyer include two integer counters: NS for *Number of Swaps* and SF for *Swap-Free* buyers; these counters NS and SF have informal semantics similar to semantics of counters NC and CF in the algorithm LSM: NS represents agent's upper estimation of number of buyers with whom it could have swaps, and, respectively, SF represents its lower estimation of number of agents that believe they have no swaps at all; in particular, the agent believes that it will not swap the current salesman with any other buyer as soon as $NS = 0$; the agent believes that there are competition for any salesman at the marketplace as soon as $NS = 0$ and $SF = 2 \times (m - 1)$, i.e. it believes that it will no swaps, and it checks twice that all buyers believe that they will not swap their salesmen. But (in contrast to LSM-algorithm) the are two options how two agents could solve whether to swap their salesmen or not.

The first is similar to *Flip_Game* in the LSM-algorithm: two agents (say i and j) play the following static game with complete information that we denote as $Swap\_Game(i, j)$:

$i \setminus j$	*decline*	*swap*
decline	$(0, 0)$	(f_i, f_j)
swap	(f_i, f_j)	(g_i, g_j)

Two moves (strategies) in this game are to *decline* and to *swap* of the salesmen. Before the game every participating agent $k \in \{i, j\}$ sends to the partner two integer values: its gain[7] g_k, if agents swap their salesmen, and f_k is its fine[8] for disagreement on declining and swapping with the partner. Again, these individual payoff values g_i and f_i are known to every agent i a priory. Then both agents compute a mixed strategy Nash equilibrium (that is a rational behavior) and play the corresponding mixed strategies until they make correlated beneficial moves. Study of this option is a topic for further research.

The second option is inspirited by SMEx-algorithm that solves MRP under the assumption of fair communication [9]. This time two agents (say i and j) also play a static game with complete information that we denote by $Coop\_Game_\alpha(i, j)$:

$i \setminus j$	*decline*	*swap*
decline	$(0, 0)$	$(0, (1 - \alpha) * g)$
swap	$(\alpha * g, 0)$	$(\alpha * g, (1 - \alpha) * g)$

In this game $\alpha, (1 - \alpha) \in (0, 1)$ represent share (concession) of the total gain $g = g_i + g_j$. In principle this value could vary from game to game, but we would like to assume that it is fixed (for the sake of simplicity) and is common knowledge for all agents. Two moves (strategies) in this game are to *decline* and

[7] In contrast to l_k in *Flip_Game*, this value could be positive as well as negative.
[8] This value must be negative as in the *Flip_Game*.

to *swap* of the salesmen. Before the game every participating agent $k \in \{i, j\}$ sends to the partner its gain g_k in the case of swap[9]. This game has a Nash equilibrium in pure strategies: (decline, decline) if $g < 0$, and (swap, swap) if $g > 0$. We call this game *Cooperative Game* since rational agents that swap their salesmen iff it is beneficial and then they share the total gain in some proportion (in contrast to *Swap_Game*, where everyone gets its own gain of loss even in the case when the total gain is positive).

Pseudocode of the SWP-algorithm is presented below. Again, we would like to comment a meaning of some variables before the description for the readability: `Me` – personal number; `Init` – initial salesman; `cur_sman, par_sman` – current, partner's salesman; `par_bel` – partner's belief that it (the partner) has nothing to swap; `contacts` – a set of buyers; `my_gain` – a gain due to swapping salesmen; `par_gain` – partner's gain due to swapping salesmen.

```
algorithm SWP::
const Me : integer in [1..m]
const Init : integer in [1..n]
var cur_sman, par_sman : integer in [1..n];
var NS : integer in [1..m];
var SF : integer in [1..2*m];
var contacts : set of [1..m];
var partner : integer in [1..m];
var par_bel : boolean;
var my_gain, par_gain : integer;
begin
1:  NS:= (m - 1); SF:= 0;
2:  cur_sman := Init;
3:  repeat
4:   if NC > 0 then NC:= (m - 1);
5:   contacts:= set of all buyers but Me;
6:   repeat
7:    partner := a buyer in the contacts ready to communicate;[10]
8:    start communication session with the partner:
9:    { send <cur_sman>, <NS=0> to partner ||
        receive <par_sman>, <par_bel> from partner }
10:   my_gain:= P_{cur_sman,Me} − P_{par_sman,Me};
11:   { send <my_gain> to partner ||
        receive <par_gain> from partner }
12:   if my_gain + par_gain > 0                    // Solve Coop_Game.
       then {cur_sman:= par_sman; NS:= (m - 1); SF:= 0}
13:    else if NS > 0
14:         then {NS:= (NS - 1); SF:= 0}
15:         else if par_bel
16:              then SF:= SF + 1
```

[9] This value could be positive as well as negative as in the *Swap_Game*.

[10] A scheduler resolves this request.

```
17:              else {NS:= (m - 1); SF:= 0}
18:    close communication session with partner;
19:   contacts:= remove partner from contacts;
20:   until contacts becomes empty
21: until (NC = 0 ∧ CF = 2*(m - 1))
end.
```

Some important properties of this SWP-algorithm are accumulated in the following two Lemmas 4 and 5. Their proofs will be published in a forthcoming full research report.

Lemma 4. *Assume that it is common knowledge in a system that all agents execute one and the same algorithm SWP, and that every agent in the system has known its initial individual salesman before execution of the algorithm. Then every agent in this system at every moment of time after the execution of line 2 always knows some individual salesman that is someone in the set of all initial individual salesmen.*

Lemma 5. *Assume that all agents execute one and the same algorithm SWP, that every agent has initial individual salesman before execution of the algorithm, and that communication is fair in the system. Then the system eventually terminates.*

Lemmas 4 and 5 together imply the following proposition.

Proposition 2.

If *a system with fair communication consists of $m > 0$ buyers each of which knows some initial individual salesman among $n \geq m$ salesmen, it is common knowledge (in the system) that all buyers are agents that execute algorithm SWP,*

then *every agent will eventually terminate, it will know upon termination an individual salesman (that is someone in the set of all initial individual salesmen), and it will know that it is impossible to reduce the total price all buyers have to pay by pairwise swapping salesmen.*

Proof of this proposition will be published in a forthcoming full research report.

In the Conclusion we demonstrate that the above algorithm SWP can not solve the graph-theoretic *assignment problem* that is (in our settings) to compute an assignment of buyers to robots with the *cheapest* total price [3]. But we do not know whether our two multiagent algorithms LSM and SWP coupled together could solve the problem or not. This topic also requires further research.

4 Conclusion

This paper is an extended and revised version of a preliminary short abstract [8]. In both papers we use *imperative* pseudocode a-lá [11] to present algorithms (protocols) LSM and SWP. It makes our agents just "executers". In contrast,

logic pseudocode (similar to [12]) is more adequate for presentation of *intelligent* agents. So, the next research topic is transition from imperative to logical pseudocode, i.e. to intelligent agents at the marketplace.

But what is more interesting for us in the case of LSM and SWP algorithms, is the following phenomenon: the common knowledge in a system of agents executing these algorithms emerges after double-check of individual beliefs. This observation leads us to the following topic for further research: When multiple (but finite and bounded) double-check of individual beliefs leads to common knowledge?

Let us repeat here also couple of other topics for further research that have been mentioned at the end of Sections 2 and 3: Whether the outcome of the LSM-algorithm is a Pareto-optimal assignment? Whether algorithms LSM and SWP coupled together could solve the assignment problem?

Unfortunately, SWP-algorithms alone cannot solve the assignment problem as follows from a the counterexample below.

$b \setminus s$	s_1	s_2	s_3
b_1	0	1	3
b_2	3	0	1
b_3	1	3	0

This table presents weighted bipartite graph that consists of the vertices $\{b_1, b_2, b_3\}$ for buyers and $\{s_1, s_2, s_3\}$ for producers. Assume also that the initial assignment of producers to consumers is $b_1 \leftarrow s_2$, $b_2 \leftarrow s_3$ and $b_3 \leftarrow s_1$. In these settings, the exchange in line 12 of the algorithm SWP cannot work and hence the buyers will terminate with the unchanged assignment, but this assignment is not optimal.

References

1. **Apt K., Grädel E.** (Eds.) Lectures in Game Theory for Computer Scientists. — Cambridge University Press, 2011.
2. **Brams S.J. and Taylor A.D.** Fair Division — From cake-cutting to dispute resolution. — Cambridge University Press, 1996.
3. **Burkard R., Dell'Amico M., Martello S.** Assignment Problems. — SIAM, 2009.
4. **Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.** Reasoning about Knowledge. — London: MIT Press, 1995.
5. **Hintikka J.** Knowledge and Belief. — Cornell University Press, 1962.
6. **Parikh R.** Social Software. — Synthese, v.132, 2002, p. 187-211.
7. **Russell S.J., Norvig P.** Artificial Intelligence: A Modern Approach (3rd ed.). — Prentice Hall, 2010.
8. **Shilov N.V., Garanina N.O.** Rational Agents at the Marketplace. — In Proceedings of the 3rd Workshop Knowledge and Ontologies *ELSEWHERE* (July 01, 2011, Novosibirsk, Russia). — A.P. Ershov Institute of Informatics Systems, 2011, p.21-28.

9. **Shilov N., Garanina N. and Bodin E.** Multiagent approach to a Dijkstra problem. — In Proceedings of Workshop on Concurrency, Specification and Programming CS&P'2010 (Helenenau, September 27 – 29, 2010). — Humboldt-Universitat zu Berlin. Informatik-Bericht Nr.237, v.1, p.73–84. (Available at `http://www2.informatik.hu-berlin.de/ki/CSP2010/proceedings2010.zip`, visited May 10, 2011.)
10. **Shilov N.V., Shilova S.O.** Etude on theme of Dijkstra. ACM SIGACT News, 2004, v35(3), p.102–108.
11. **Tel G.** Introduction to Distributed Algorithms. — Cambridge University Press, 2nd Edition, 2000.
12. **Valiev M.K., Dekhtyar M.I., and Dikovsky A.Ya.** Systems of Agents Controlled by Logical Programs: Complexity of Verification. Programming and Computer Software, 2009, v.35(5), p.266-281.
13. **Wooldridge M.** An Introduction to Multiagent Systems. — John Willey & Sons Ltd, 2002.