Effective Interaction in Asynchronous Multi-Agent Environments for Supply Scheduling in Real-Time

Alexander Tsarev
Knowledge Genesis Group Ltd.
Samara, Russia
email: mail@identifier.at

Petr Skobelev
Samara State Aerospace University
Samara, Russia
email: petr.skobelev@gmail.com

Dmitry Ochkov
Smart Solutions Ltd.
Samara, Russia
email: ochkov@smartsolutions-123.ru

***Abstract*—This paper focuses on analysis of effective interaction techniques of agents in multi-agent systems used for real-time scheduling. The paper describes two approaches to the organization of the interaction of asynchronously working software agents. The supply network scheduling case is considered to show the difference in how the interaction goes on. The comparison shows how well each approach allows parallel processing, and subsequently, how fast the scheduling can be done on multi-core hardware. The pros and cons of the approaches are described, as well as ways to achieve better quality. Finally, the results of processing of real data using the approaches are given. The results show a higher effectiveness of one of the approaches in real-time supply scheduling.**

Keywords—real-time; scheduling; software agent; multi-agent; supply chain; supply network; supply demand; interaction protocol; agent negotiation; asynchronous interaction; processing speed; parallel processing; schedule quality.

I. INTRODUCTION

Growing complexity and dynamics of modern global market demand new paradigms in resource management [1][2]. New revolutionary approach to increase efficiency of business is associated today with real-time economy, which requires adaptive reaction to events, ongoing decision making on resource scheduling and optimization and communication results with decision makers.

Multi-agent technology is considered as a new design methodology and framework to support distributed problem solving methods in real-time scheduling and optimization of resources [3][4].

The main feature of real-time scheduling and optimization methods is to produce a result in the specified moment of time or time interval, reacting to unpredictable external and internal, constructive or destructive events (new order coming, order is cancelled, resource unavailable, etc.).

The quality and efficiency of decision making in resource scheduling and optimization process can be influenced by the number of factors: the intensity of events flow, the number and current state of resources, individual specifics of orders and resources, time interval between the events and processing time for events, productivity of resources and many others.

A big challenge is to ensure that certain quality of scheduling results is achieved in a short time after the event to make it possible to finish the processing before the next event and to always have a valid schedule needed for decision-making.

Figure 1 illustrates the difference in actuality of scheduling results (how well they reflect reality) in the changing environment. Having frequent data updates, it becomes more important to process them faster to get a valid result (green line). Otherwise, one can use a lengthy processing to get an optimal result (yellow/red line), but this result does not consider the last changes. Then, we are forced to always base your decisions on an optimal, but outdated picture.

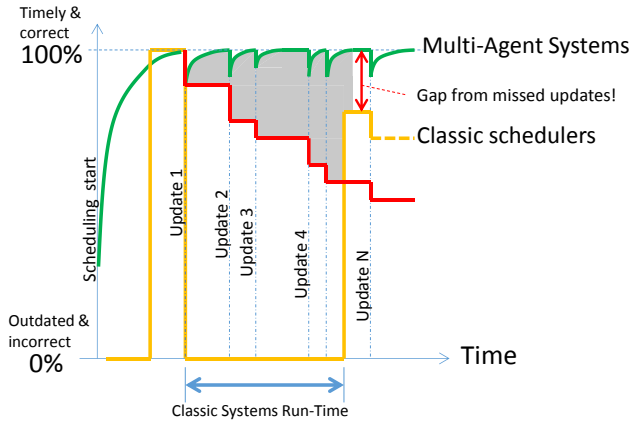

Figure 1. Real-time adaptive scheduling results.

One of the main problems of classical methods and algorithms [5][6] is that complexity of scheduling with new criteria grows exponentially. This makes their applications very limited in practice. Many heuristic methods allow obtaining close to optimal solution within a reasonable time. Hybrid heuristic algorithms integrate traditional dispatching rules with genetic, neural, swarm and other approaches. Obvious disadvantages of the centralized methods of scheduling and optimization resource management lead to development other approaches, in particular distributed problem solving methods. Bio-inspired evolutionary (genetic and swarm) algorithms are applied both in centralized and decentralized systems [7]-[9]. They have proved to be more useful, reliable and generic scheduling and optimization tool for business. One of new approaches is based on bio-inspired distributed problem solving of resource scheduling problems based on multi-agent technologies with economic reasoning. This approach can combine benefits of bio-inspired, DCOP and virtual market methods based on multi-agent technology

 ISBN: 978-1-61208-405-3

and is designed to support self-organization of schedules to provide flexibility in event processing. Multi-agent resource allocation is used for job scheduling and some other tasks [11]. In our paper, we consider a more specific practical case of supply scheduling and compare the interaction approaches from the perspective of their use in real-time application. There are other researches done regarding the use of multi-agent approach in supply chain scheduling, including analysis of high-level protocols (Combinatorial Auctions, Bargaining Processes, Random Search, Knowledge Based Systems, Learning Systems) [12][13], but they do not focus on the analysis of benefits of different agent assumptions in asynchronous environment.

To solve the problem of multi criteria scheduling and optimization it is suggested to use Demand-Resource Network concept (DRN) based on holonic approach and compensation method for real-time resource management on a virtual market [10]. In accordance with this distributed approach, initial complex problem is decomposed into more simple and specific problems – to schedule and optimize orders, resources and products with the use of demand and supply agents. All agents are working continuously trying to maximize their objective functions and use money to solve conflicts by negotiations and finding trade-offs (until local optimum is reached or time is expired) with compensations in case that some of them change position losing money.

Objectives, preferences and constraints of agents are defined by individual satisfaction functions and bonus/penalty functions. As the result of agents decision making, a local balance is reached and situation when no agent can change position is recognize as a consensus which stops computations. As a result, the solutions (the schedule of resource usage) comes not from one algorithm but evolves (emerges) dynamically in process of agents interactions and negotiations. Solution search and adjustment process stop when the consensus is found or when the time limit is exceeded for finding a solution, and if not the whole - but partial problem solution will arrive that will be interactively finalized by the user.

The use of multi-agent approach provides many potential benefits and possibility to speed up the scheduling by use of parallel processing of asynchronous agents. Still, this possibility depends on how the agents interact with each other and on their dependence on each other in decision making. Obviously, the scheduling task requires a lot of information to be transferred between the agents to allow a better search for result. This transfer not only takes time itself, but also may force the agents to wait each other. In this paper, we consider two fundamental approaches to agent interactions related to the question when the agent should ask or wait for information, and when it can make independent decisions.

In Section II, we describe what approaches to agent interactions we consider in this paper. In Section III, we compare the interaction schemas based on particular supply routing example. In Section IV, we show how the lack of resources in the supply network affects the interactions, performance and quality of results, and propose the ways to mitigate the drawbacks. In Section V, we compare the approaches based on a more complex case of competing orders in supply network. In Section VI, we provide the results of comparison based on real supply network data, including the difference in performance and quality of the approaches. In Section VII, the conclusion is given.

II. APPROACHES TO AGENT INTERACTION IN SUPPLY SCHEDULING

In this paper, we compare two different approaches to the organization of multi-agent interaction in relation to the supply scheduling. One approach is based on request and reply and follows the rejection presumption principle, which means that if no reply is given it is an equivalent of rejection (sender must wait for an answer). This approach is referred to as rejection assumed interaction in the paper. Another approach is based on the acceptance presumption principle, which means that without explicit rejection from the counterpart of communication the acceptance of request is assumed. This approach is referred to as acceptance assumed interaction in the paper. Of course, this relates to the requests that do not require an informational feedback, but only ask another site to do something, while the feedback is optional.

Let us consider the difference based on a simple example of a network consisting of one shop and two storages that can supply it (Figure 2 and Table I).

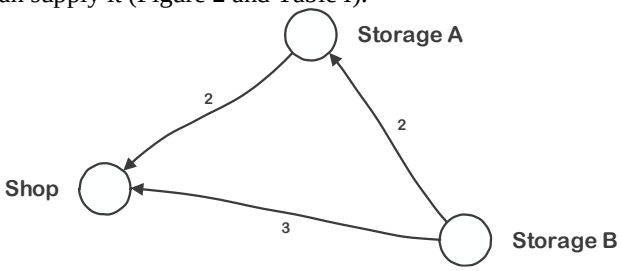

Figure 2. Example of supply network.

There is an order at the Shop for one item of Product. Transportation costs are listed in Table I, and there are no other costs.

TABLE I. TRANSPORTATION COSTS

Source	Destination	Cost per item
Storage A	Shop	2
Storage B	Shop	3
Storage B	Storage A	2

In the simplest example, we have enough stock at both storages. The rejection assumed interaction looks as the following, in this case (Figure 3).

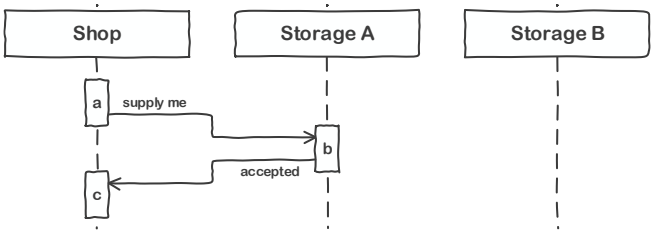

Figure 3. Interaction based on rejection presumption.

It is an obvious case. The order at the shop requests the cheapest channel (channel from Storage A costs 2 while the other channel is 3) if the Product can be delivered and gets the positive reply. The interaction takes three steps in total. Two of them (‘a’, ‘b’) are time consuming, as they may require some analysis, while ‘c’ does nothing, but still takes some time to initialize the agent and process the message. For the sake of simplicity, let us decide that steps with analysis take 1.0 time unit, while steps without significant data processing take 0.1 time units (tu). In this case, the total is 2.1 tu.

If we consider the acceptance assumed interaction for this case, the only difference is that we do not need the last step (Figure 4), as we assume the request is accepted and the supply is possible. Therefore, the total time for processing is 2.0 tu.

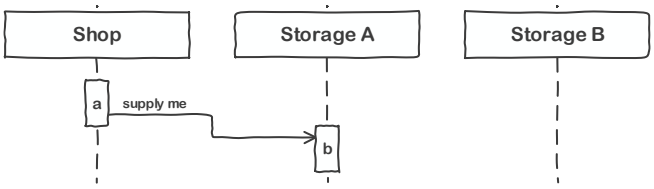

Figure 4. Interaction based on acceptance presumption.

It is important to note that even if an additional processing is needed at the shop to obtain a final result (schedule) after the supply request is considered accepted, in acceptance presumption case this happens immediately after the request is sent and does not take additional time, as being done in parallel with the request processing at the storage.

III. AGENT INTERACTION IN SUPPLY ROUTING SCENARIO

Now, let us consider a less trivial case, where Storage A is empty. The order at the Shop does not have this information and still asks it first in the hope to get cheaper supply. This leads to the following sequence of interactions.

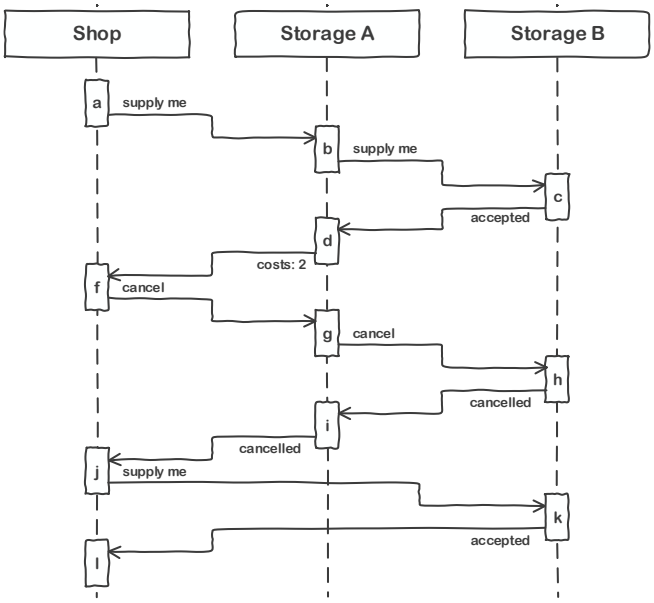

Figure 5. Routing based on rejection presumption.

Storage A, in this case, at step ‘b’, cannot fulfill the request and sends a supply request to Storage B. At step ‘c’, Storage B reserves the stock (creates its own schedule) and sends the acceptance. Then, on step ‘d’ the Storage A sends the acceptance with the additional cost of transportation from B to A. This actually tells the order at the Shop that the total cost of supply will be 2 (from A to Shop) + 2 (from B to A) = 4. This is more than the cost of transportation from B to Shop, which is 3. This makes the order to try another channel. It cancels the previous request (in order to let A and B update their schedule and free the reserved stock) and asks the Storage B directly. The whole interaction takes 11 steps, with four of them (‘d’, ‘i’, ‘j’, ‘l’) being just fast reply processing. Therefore, the total scheduling time is 7.4 tu. It might be unclear why the steps ‘j’ and ‘l’ are “short”. It is because we consider the scheduling process at Shop to be almost completely done at step ‘f’. When Shop gets the cost reply from Storage A, it has to re-build the schedule to be supplied from Storage B. This may happen in slightly different ways across the steps ‘f’, ‘j’, and ‘l’, but the total re-scheduling time at Shop is assumed to be 1 tu in average, and we just associate this time with the step ‘f’.

If we use the acceptance assumed interaction, we get a significantly different picture (Figure 6).

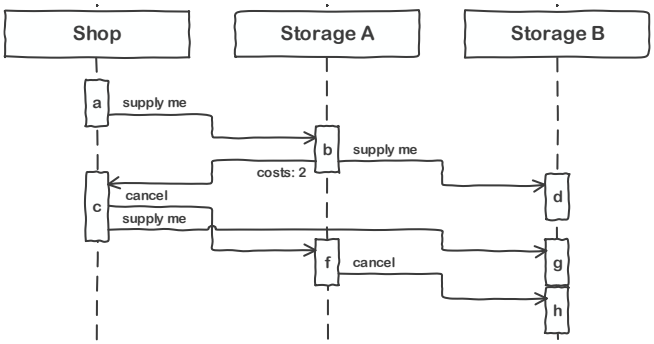

Figure 6. Routing based on acceptance presumption.

We have 7 steps here in total, but all of them are time consuming. More important is that the messages are sent to several recipients at steps ‘b’ and ‘c’ as we do not wait for reply, and the corresponding sites process them in parallel. The steps ‘c’ and ‘d’ go in parallel, as well as ‘f’ and ‘g’. This allows packing of all the 7 steps into 5.0 tu instead of 7.4 tu of synchronous interaction.

It is important to note that rejection assumed interaction does not mean synchronous processing (scheduling, in our case). There are still things you can do in parallel. For example, interactions happening in different parts of the network can go in parallel. However, with the increasing number of events to be processed also the likelihood of touching the same site increases. If this happens, we need to wait until the first event is processed completely.

IV. AGENT INTERACTION IN RESORCE DISCOVERY SCENARIO

However, there is a drawback in acceptance assumed interaction, which is clearly seen on the following example. Let us take the case, where the stock at the Storage B is

 ISBN: 978-1-61208-405-3

limited to just one item. If we use rejection presumption, the scheduling process and the result are exactly the same as if the stock is not limited, which is correct. But, if the acceptance presumption is used, we get the following interactions (Figure 7).

At step ‘g’, it happens that the one item in the stock of the Storage B is reserved for the Storage A. The reservation takes place at step ‘d’, and the cancellation is now on its way, but the direct request comes faster due to asynchronous nature. This forces the Storage B to reject the direct request from the Shop. Consequently, the Shop asks Storage A again. This not only adds the steps to the interaction and 2.0 tu of processing time, but also leads to the non-optimal result. The Shop is supplied, but the cost is higher, because it is supplied indirectly.

This non-optimality can result in a lower quality of the final schedule, but it happens only in the specific situation when there is lack of stock and re-negotiation between sites takes place at the same time. Even if rejection presumption is used, similar situation still can happen when several orders compete asynchronously. In practical cases, the lack of stock affects a very small part of the orders and the decrease in scheduling is normally acceptable, as the whole scheduling process does not normally achieve a global optimum. Still, this decrease can be a problem in some cases.

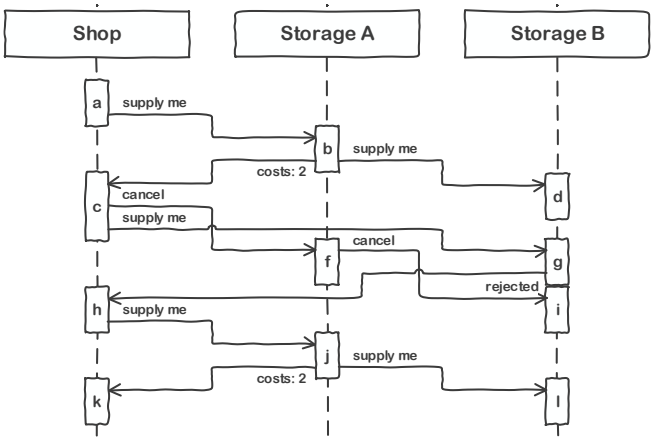

Figure 7. Resource discovery based on acceptance presumption.

Fortunately, there are ways to avoid this problem. One of them is to postpone rejection. With this approach, the site requested for supply does not send the rejection immediately, but waits for the specific number of steps (to be decided later). If the rejection is still needed after several steps (no other requests were cancelled) – it is sent. If we do so in the last case, the rejection is not sent on step ‘g’, and on step ‘i’ it is no longer needed. The scheduling goes in the same way as in the unlimited stock case (no rejection is sent because the cancellation comes at step ‘h’) taking 5.0 tu and optimal result (Figure 6). The weakness of this method is in the necessity to specify the number of steps to postpone the rejection. The number should be high enough to let all the asynchronous cancellations to come before the rejection should be sent. This depends on the structure of the network. And, in the cases when the request really should be rejected the postponement increases the total scheduling time by the number of the specified postponement steps. But the rejection postponement is needed only when the stock is short, which is less than 5% of the requests during the scheduling and does not affect the total timing significantly.

Another method is to track the final product recipient (root order) in the messages so that the site (Storage B, in our case) knows whom the stock is reserved to. If it gets a new request for the same root order, the previous reservation is cancelled automatically. This method also works perfectly and does not depend on the network structure. But the need to check the reservations in the stock for specific order increases the scheduling time at the site. We use this method in practice due to its simplicity and reliability.

V. AGENT INTERACTION IN COMPETING ORDERS SCENARIO

The next two examples concern the comparison of the interaction protocols in the situation when there are several competing orders in the network. Taking the case where we have empty stock at Storage A and sufficient stock at Storage B, let us introduce one more sales point in the network to make it look like on the following picture. Now, we have an order at each of the two shops (Figure 8 and Table II).

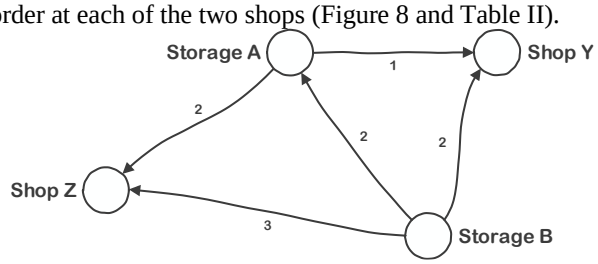

Figure 8. Supply network with two competing orders.

TABLE II. TRANSPORTATION COSTS IN THE NETWORK WITH TWO COMPETING ORDERS

Source	Destination	Cost per item
Storage A	Shop Z	2
Storage B	Shop Z	3
Storage B	Storage A	2
Storage A	Shop Y	1
Storage B	Shop Y	2

Following the rejection presumption principle, the sites cannot process next request until they get a response from other sites regarding the previous request. Thus, in our case the Storage A becomes a bottleneck because both shops ask it first (as potentially cheaper source), and it cannot answer them both until Storage B answers the request. For example, the request processing is blocked at the Storage A on the steps ‘c’, ‘d’, and ‘e’ on the following diagram (Figure 9), which leads to the delay of the processing of the request from the Shop Y on step ‘g’.

Specifically, when Storage A gets a request from Shop Z at step ‘c’, it sends a request for this product to Storage B (as it does not have it in the stock). When the request from Shop Y comes (almost the same time as from Shop Z), it cannot be processed until the request to Storage B is accepted.

 ISBN: 978-1-61208-405-3

We consider only one product in the network in the paper, so the orders compete for the same stock. If you get requests for different products, they theoretically may be processed immediately one after another, but this is an abstract situation. In practical tasks there are much more interdependencies between resources and demands other than just product type. For example, the channel capacity between Storage A and Storage B, or the dispatch capacity at Storage B can be limited, or the transportation cost may depend nonlinearly on the volume transported. This prevents Storage A from answering the second request even if it is for a different product, until the acceptance of the first delivery is received (or assumed).

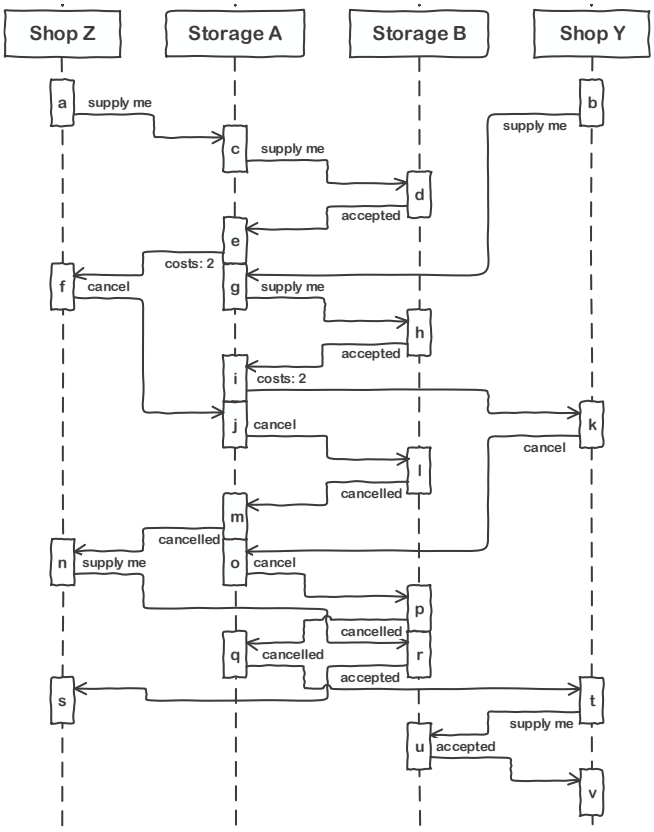

Figure 9. Competitive interactions with rejection presumption.

The complete processing of the two orders with this approach takes 22 steps. Considering that some of them are done in parallel and some of them are very quick, this exact sequence takes 12.4 tu.

Actually, we do not consider here the fully synchronous interaction that requires all events to be processed separately. It means that the order from the Shop Z is completely processed first, and only then the processing of the order from the Shop Y starts. This forces the whole sequence to go in one thread and take 16.6 tu.

The next diagram (Figure 10) shows the interactions using acceptance presumption protocols.

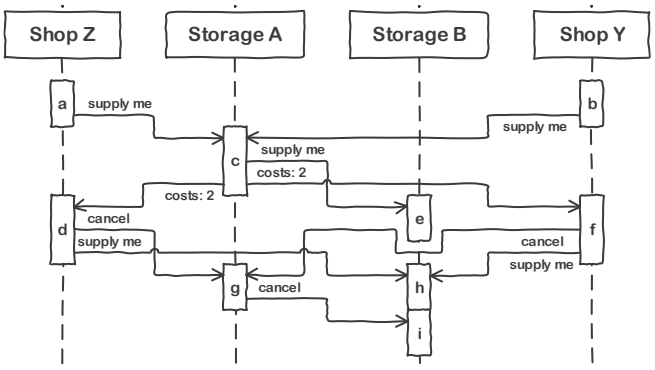

Figure 10. Competitive interactions with acceptance presumption.

One can see that, in this example, the structure of the interactions is the same as in the case where we had only one order. The whole process goes as much as possible in parallel and takes the same 5.0 tu. The significant difference from the rejection presumption case is that at some steps several requests are processed by the site simultaneously. From one point of view, such steps should take more time, but from the other point of view, the processing of several requests at once never takes more time than separate processing of the same requests. What is more important, having several requests at once allows avoiding blind decisions that should be re-considered when the next request comes. A separate paper is dedicated to this phenomenon.

VI. COMPARISON BASED ON REAL DATA

Thus, based on the examples above, the theoretical comparison of the two approaches is shown in Table III.

TABLE III. EXAMPLES SUMMARY

Case	Fully synchronous processing	Rejection presumption (tu)	Acceptance presumption (tu)
One-level depth interaction.	2.1	2.1	2.0
Two-level depth interaction without resource constraints.	7.4	7.4	5.0
Two-level depth interaction with resource constraints.	7.4	7.4	7.0
Two orders, two-level depth interaction.	16.6	12.4	5.0

The practical cases are much more complex in terms of the depth of interactions as well as of the number of events processed in parallel. We used a real client data including more than 300 sites in the network (part of which is fully interconnected) and about 10 000 orders to model different interaction protocols. The network to be scheduled includes several factories, their storages that can interexchange materials and final products, and customer distribution centers that should be supplied. The model also includes production scheduling and some other features that affect the processing time in different situations. The modelling has been done using 16-core processor. Table IV presents the results of the modelling.

TABLE IV. REAL DATA PROCESSING

	Processing time (ms)	Messages between sites	Achieved quality ($)
Fully synchronous processing	737236	3200	1813499
Rejection presumption	191334	3140	1813359
Acceptance presumption	50275	2333	1812240

The slight difference in quality between the synchronous processing and the rejection presumption most probably happens because of asynchronous stock competition between different orders.

Comparing the last two rows we can see that the use of acceptance presumption approach gives us 3.8 times faster processing and decreases the quality by about 0.1%, which seems to be a fair price in most cases.

VII. CONCLUSION

The acceptance assumed interaction works much better than the rejection presumption in multicore and especially in distributed environments because waiting for reply there is especially costly. However, it is fragile in non-reliable communication environments. If the requested site in the network does not implement the request and does not send the rejection, the requesting site works in wrong assumptions and the whole schedule is not consistent. This is why it can only be used within well-communicated infrastructure, normally related to one company.

We use the acceptance presumption approach in the industrial applications for supply networks management.

It is also important to make a research how the two approaches can be combined in some way during the interaction. Although the acceptance presumption looks working better in most cases, especially below the resource limits, which is over 90% of the practical cases, the rejection presumption may still allow getting results of higher quality without using workarounds in the low resource situations.

ACKNOWLEDGMENT

This work was carried out with the financial support of the Ministry of Education and Science of the Russian Federation.

REFERENCES

[1] G. Rzevski and P. Skobelev, Managing complexity, 2014, WIT Press, Boston.

[2] A. Park, G. Nayyar, and P. Low, Supply Chain Perspectives and Issues, A Literature Review, April 21, 2014, Fung Global Institute and World Trade Organization.

[3] A. Mohammadi and S. Akl, Scheduling Algorithms for Real-Time Systems, Technical Report, 2005, no. 2005-499, School of Computing Queen's University, Kingston.

[4] M. Joseph, Real-time Systems: Specification, Verification and Analysis, Prentice Hall, 2001.

[5] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2008.

[6] J. Leung, Handbook of Scheduling: Algorithms, Models and Performance Analysis, CRC Computer and Information Science Series, Chapman & Hall, 2004.

[7] S. Binitha and S. Sathya, "A survey of bio inspired optimization algorithms", Int. Journal of Soft Computing and Engineering, 2012, vol. 2, issue 2, pp. 2231-2307.

[8] S. Sun and J. Li, "A two-swarm cooperative particle swarms optimization", Swarm and Evolutionary Computation, 2014, vol. 15, pp. 1-18. Elseiver.

[9] M. Tasgetiren, M. Sevkli, Y. Liang, and M. Yenisey, "Particle swarm optimization and differential evolution algorithms for job shop scheduling problem", International Journal of Operational Research, 2008, vol. 3, no. 2, pp. 120-135.

[10] V. Vittikh and P. Skobelev, "Multiagent interaction models for constructing the demand-resource networks in open systems", Automation and Remote Control, 2003, vol. 64, issue 1, pp. 162-169.

[11] Y. Chevaleyre, et al. "Issues in Multiagent Resource Allocation", https://staff.science.uva.nl/u.endriss/MARA/mara-survey.pdf, retrieved: March 2015.

[12] M. Barbuceanu and M. S. Fox, "Coordinating multiple agents in the supply chain", Proceedings of the fifth workshops on enabling technology for collaborative enterprises, WET ICE'96, IEEE Computer Society Press, 1996, pp. 134-141.

[13] T. Stockheim, M. Schwind, O. Wendt, and S. Grolik. "Coordination of supply webs based on dispositive protocols", 10th European Conference on Information Systems (ECIS), Gdañsk, 6-8 June 2002.

[14] A. Oliinyk, "The multiagent optimization method with adaptive parameters", Artificial Intelligence journal, 2011, no. 1, pp. 83-90.

 ISBN: 978-1-61208-405-3