Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers in Industry 54 (2004) 273–290
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Application of multi-agent planning to the assignment problem

S. Kornienko[a,*], O. Kornienko[a], J. Priese[b]

[a]*Institute of Parallel and Distributed High-Performance Systems, University of Stuttgart, Universitätsstr. 38, Stuttgart D-70569, Germany*
[b]*Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Seidenstr. 36, Stuttgart D-70174, Germany*

Received 16 October 2002; accepted 29 November 2003

Available online 3 March 2004

Abstract

Nowadays, a globalization of national markets requires developing the flexible and demand-driven production systems with new innovative concepts of management, information processing, production scheduling and planning. The presented work focuses on the low-level planning, where the multi-agent solution towards a ''job-machine'' assignment is considered. The main point of the discussion is the flexibility of planning systems ensured by the concept of agent's ''roles'' and ''emergencies''. Depending on the state of ''emergency'', the system receives stepwisely additional degrees of freedom to adapt the planning to the changing conditions of the manufacturing floor. The distributed constraint satisfaction and optimization approaches, underlying the suggested method, as well as activities of rescue agents, are described in the form of Petri networks providing both the conceptual notions and main details of implementation.

Keywords: Flexible manufacturing system; Process planning; Multi-agent systems; Petri networks; Distributed optimization

1. Introduction

In the modern world, competition among international suppliers and globalization of national markets requires production systems that can successfully operate in this global and quickly changing market (e.g. [1]). From this viewpoint there are several requirements that should be satisfied by these systems. Firstly, the time needed from development of a product to its serial production should be essentially shortened. Secondly, manufacturing systems should become oriented to a multitude of parts and variants of customer requirements. This means a product will be fabricated in small series with different consumer properties, like color, equipment and so forth. Moreover, a product should often be fabricated on the demand of a client with a unique specification [2], achieving the aim of mass customization [3]. All these requirements may only be satisfied by flexible, quickly reconfigurable production systems. Not only the physical fabrication should be flexible (equipped with e.g. reconfigurable machinery), but also all operational, executive and developing processes of modern production systems. For such a factory a completely new structure, new organizational principles and, correspondingly, new software and hardware instruments should be developed [4].

Taking into account a spatial distribution of manufacturing elements and the requirement of flexibility to the whole system, the concept of autonomous

* Corresponding author.
E-mail address: korniesi@informatik.uni-stuttgart.de (S. Kornienko).

0166-3615/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2003.11.002

agents has found some applications in this field [5]. Moreover, the following applications have been taken into account:

- Activity of agents is a result of the group behaviour that bases on different forms of negotiations among agents. Because of this specific form of ''programming'', the problem solving (decision making, planning, etc.) in a multi-agent system (MAS) has essentially more degrees of freedom, than in traditional centralized systems. The negotiation-based MAS planning system becomes more flexible and, in this way, more ''stable'' to different predicted and (sometimes unpredicted) disturbances, e.g. machine failures, technological changes etc.
- There is a trend to equip processing elements (processing machines) with some degree of autonomy and ''intelligence'', allowing them to react to short-term disturbances, perform self-maintenance and integrate to autonomous manufacturing. This trend corresponds to the agent concept, in this way the processing element becomes an agent.
- In some situations (e.g. hazardous and dangerous environments) a human worker needs to be replaced in modern manufacturing. The replacement element should have a behaviour similar to human, i.e. it should be autonomous, make decisions and communicate with human or non-human workers.

Application of the agents to manufacturing requires also a development of new approaches towards typical problems of multi-agent technology, such as distributed problem solving, planning or collective decision making [6]. The following agent-oriented application is addressed to the lowest level of manufacturing architecture, where the low-level jobs (for example ''to produce one workpiece with a defined specification'') should be assigned to available machines. The aim is to generate this assignment via agents that represent different factory departments as well as processing elements.

The remainder of this work is structured in the following way. Section 2 describes the assignment problem from the manufacturing side. This section is concluded by a formulation of constraints needed for the next steps. Section 3 is devoted to the agent-oriented solution of the assignment problem. The main point of this section is focused on the flexibility of MAS and on the methods that allow it to be achieved. Several remarks about agent-based optimization are brought out in Section 4. Finally, two issues concerning disturbances and executing the planning approach are summarized in Section 5.

2. The problem of assigning jobs to machines

The assignment problem is often encountered in manufacturing. It is a part of Operations Research/Management Science (OR/MS), where the flow-shop and job-shop problems with deterministic, stochastic, one-step or many-steps characters are distinguished [7]. Generally, the assignment problem can be classified into scheduling, resource allocation and planning of operation order (e.g. [8]). This is a classical *NP*-hard problem, there are known solutions by combinatorial optimization [9], dynamical optimization [10], evolutionary approaches [11], constraint satisfaction and optimization [12] as well as discrete dynamic programming [13]. However, these methods are developed as central planning approaches, the distributed or multi-agents planning for the assignment problem has in fact not been investigated (overview e.g. in [14]).

Manufacturing of a workpiece consists of different steps and requires a corresponding plan that is known as process planning. The process plan (PP) includes a technological working plan and manufacturing control. The first from them defines how to manufacture the workpiece whereas the second determines available machines and production timing. Because of the separation between these branches, especially because of their sequential execution, today's concepts for process planning are not able to react reasonably to disturbances like machine malfunctions or changes of production order. One way to solve this problem is to develop a rescheduling tool allowing a worker to adapt the generated plan to the actual situation on the shop floor. A very promising concept exists that is based on an integrated information model, which enables rescheduling in the case of disturbances or plan variances [15]. In order to make the process planning more adaptable and to improve the reaction time to disturbances, it is reasonable to incorporate both parts of PP into a so-called sequence plan.

Implementation of the sequence plan needs a new approach to process planning in order to better adapt a

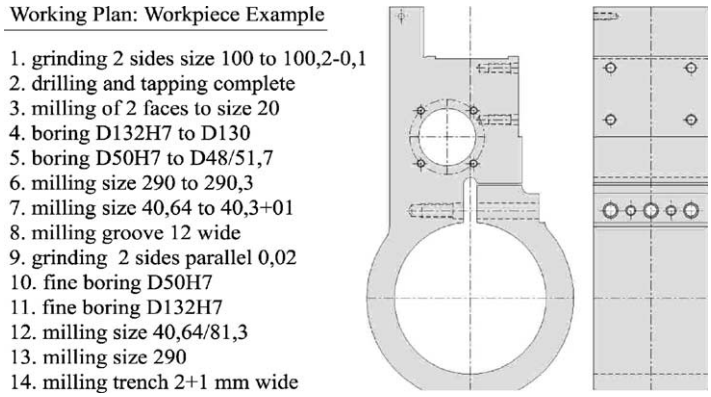

Fig. 1. Working steps for manufacturing the workpiece B (see Table 1).

schedule to the real situation. This is possible by using a new concept for the integration of manufacturing control and working plans. Here we introduce the hierarchical working plan that enables a plan to be adapted on the base of a rough plan and the dynamic working plan based on the short-ended fine plan. Then the mentioned new concept represents a composition between the dynamic working plan and the hierarchical working plan and leads to an appropriate sequence plan. Moreover, a hierarchical structure of a working plan enables a necessary adaptation of the production capacity.

Because an assignment or scheduling of working steps and machines has to be done in a short-ended time, an implementation of this concept requires a high level of automation. It can be thought of as an assistance tool for a skilled worker for generating the time- or/and cost-optimized sequence plan. In this context an agent-based planning approach seemed to be promising.

As described above, the mentioned concept of integration has to be decomposed into subtasks. However, there are two important steps concerning the decisions made. On the one hand, is the decision of whether the machine is technologically able to manufacture (a part of working planning) and, on the other hand, whether the machine is organizational available, e.g. it is without prearrangements. If there is more than one machine that satisfies these boundaries, the obtained scheduling can also be optimized.

To prove the technological ability of a machine, the technical, technological and geometrical criteria for a workpiece have to be verified. Technical criteria determine necessary features of a workpiece and on this basis it can be decided whether a machine is able to manufacture this kind of feature, e.g. the drilling feature. Technological criteria determine whether the machine can operate with a necessary quality or determine a technologically necessary order, e.g. the statutory tolerances. Geometrical criteria result from a geometrical description of the workpiece, for example whether a necessary clamping is possible.

Fig. 1 shows an exemplary working plan with the corresponding working steps and a drawing of the workpiece. Technological criteria in this example are e.g. the fine-boring after boring (working step Nos. 10 and 11 after Nos. 4 and 5), the milling groove after boring (No. 8 after No. 4) or the milling trench after the milling size 290 (No. 14 after No. 13).

Organizational criteria are a set of specifications of production orders and machines. Firstly, it defines whether a machine has an available time slot and, secondly, whether this time slot is suitable to manufacture the given production order. All these criteria determine the agent-based process planning in the form of corresponding constraints, which reduce the decision space in the assignment problem. For optimization, the cost criterion, based on the simplified cost model, is considered. Moreover, all machines have the same overheads. So it is possible to reduce cost by a prevention or reduction of transportation of a workpiece. Therefore, manufacture of a piece on one available machine for as long as possible can be considered as one optimization factor.

The following scenario is based on a flexible manufacturing system (FMS). This kind of production

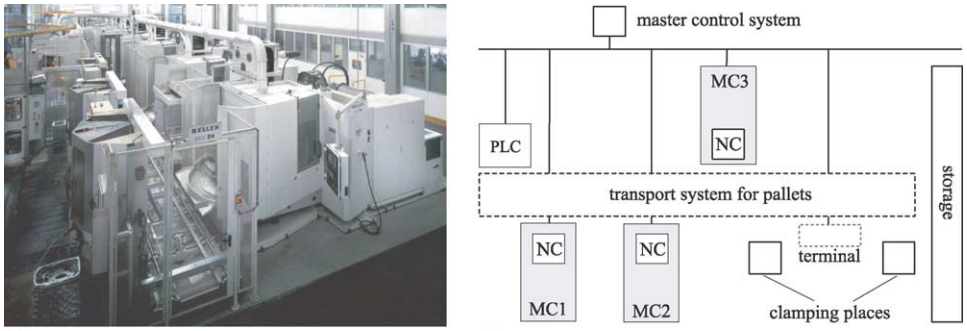

Fig. 2. (a) Example of a real flexible manufacturing system (FMS) (courtesy of Heller Maschinenfabrik GmbH); (b) Layout of FMS used in the presented work, where NC is a numerical control, PLC is a programmable logic controller, and MC is a machine center.

system is especially suitable for implementation of the agent-based process planning as a consequence of the high level of flexibility and technological abilities of FMS. FMSs are the networks of machines with some or completely equal abilities, being able to manufacture the same production tasks. They are also able to process continuously and synchronously several workpieces in different manufacturing processes without retooling.

The typical design of a FMS is shown in Fig. 2 which exemplifies the presented approach. The main properties are three machining centers, a transport system, clamping places, buffer places and storages for workpieces and tools. Additionally, the system possesses control and communication parts. In this example, all machines have partly equal abilities, but no machine is suitable to handle all production orders on its own.

The technical criteria in this approach and the corresponding constraints are the following: the machine MC1 is able to drill, to mill and to turn. The machine MC2 is able to drill, to mill and to grind. The machine MC3 is able to mill, to turn and to grind. Geometrical criteria are neglected. This means that it is possible to clamp all kinds of workpieces. Technological criteria are given for every workpiece in terms of orders. All these criteria determine the technological constraints. The following approach does not contain all the technological constraints in order to reduce the complexity.

The description of these constraints is a part of the process engineering. Today, for process planning the personal knowledge base of engineers is used. However, for the future it is reasonable to automate this procedure. The suitable basis can be given by e.g. a development of *ISO14649* that represents a macroscopic metamodel for process planning [16]. Therefore an integrated planning process from design till working plan is conceivable. Such a kind of process still remains a big field of research.

In the considered example the FMS has to manufacture a multitude of parts and variants of production orders (see Table 1). In total there are five types of workpieces (5–20 pieces of each type) that have to be manufactured on available machines. Table 2 shows the sequence of working steps for the workpiece of type A, where all mentioned technological constraints are already considered.

Processing of each workpiece consists of several working steps (defined by a technological process), all these working steps cannot be processed on one machine. Each from the working steps has different length and also cost. Moreover, each type of workpiece has its own technology, i.e. its processing consists of different working steps. For simplification it is

Table 1
Types, pieces and machines in the assignment problem

Piece type	No. of pieces of each type	Technology/No. of steps	No. of availability of machines
A	5	Table A/11	3
B	15	Fig. 1/14	3
C	10	7	4
D	20	10	2
E	5	5	3

Technological restrictions required for workpieces of types A and B are shown in Table 2 and in Fig. 1 correspondingly.

Table 2
Technological Table A for workpiece type A

Working step	Length/ machine 1	Length/ machine 2	Length/ machine 3	Order
1	1	0	1	1
2	2	0	2	2
3	0	1	1	2
4	3	0	3	2
5	1	1	0	2
6	2	2	2	2
7	0	1	1	3
8	3	0	3	4
9	2	0	2	4
10	1	1	0	4
11	1	1	1	4

Zero in a length (of a working step) at corresponding machine means that this machine cannot perform the requested operation. Order of working steps means, that e.g. the steps 2–6 should be produced after the step 1 and before the step 7. It is natural to assume these steps cannot be performed at the same time on different machines.

assumed that available machines are of the same type, therefore the cost and length of the same working step do not differ on these machines (in the general case they are different). The aim is to generate a plan of how to manufacture these workpieces with minimal cost, minimal time (or other optimization criteria), taking into account the restrictions summarized in Tables 1 and 2. Because of these restrictions, this problem belongs to the so-called constraint class of problems, where, firstly, an optimization landscape is discrete (small islands on the landscape), secondly, there is no continuous gradient. As suggested by some authors, this problem can be separated into a constraints satisfaction problem (CSP) and constraints optimization problem (COP). In the given work, this methodological way is used.

Let us denote a working step as WS^i_j, where i is the type of workpiece and j the number of the working step. An available machine is denoted as M_k, where k is the number of machine. We also need to introduce a piece P^m_n, where m is the priority of production and n is the number of this piece. In this way, $\mathrm{st}(P^m_n(WS^i_j))$, $\mathrm{fn}(P^m_n(WS^i_j))$ denotes the start and end positions of the corresponding working step that belongs to the corresponding piece ($\mathrm{st}(WS^i_j)$, $\mathrm{fn}(WS^i_j)$ for all pieces). We start with the definition of these values

$$P^{m\in[1-20]}_{n\in[1-20]}(WS^{i\in\{\mathrm{A,B,C,D,E}\}}_{j\in[1,\ldots,11]}) = o \in \text{operation}, \quad (1)$$

$$M_{k\in\{1,2,3,4\}} = \{o \in \text{operation}\}, \quad (2)$$

$$\mathrm{st}(P^m_n(WS^i_j)) = \{t \geq 0, t \in R\}, \quad (3)$$

$$\mathrm{fn}(P^m_n(WS^i_j)) = \{\mathrm{st}(P^m_n(WS^i_j)) + \mathrm{length}(P^m_n(WS^i_j))\}. \quad (4)$$

The first constraint determines a correspondence between operations of working step and of the k-machine

$$C_1 = \{(o_1, o_2) | o_1 \in P^m_n(WS^i_j), o_2 \in M_k, o_1 = o_2\}.$$

The technological restrictions given by the Table 2 (for all workpieces of type A) can be rewritten in the following form:

$$C_2 = \{(\mathrm{fn}(WS^{\mathrm{A}}_{[1]}) < \mathrm{st}(WS^{\mathrm{A}}_{[2-6]})) \subset WS^{\mathrm{A}} \times WS^{\mathrm{A}}\}, \quad (6)$$

$$C_3 = \{(\mathrm{fn}(WS^{\mathrm{A}}_{[2-6]}) < \mathrm{st}(WS^{\mathrm{A}}_{[7]})) \subset WS^{\mathrm{A}} \times WS^{\mathrm{A}}\}, \quad (7)$$

$$C_4 = \{(\mathrm{fn}(WS^{\mathrm{A}}_{[7]}) < \mathrm{st}(WS^{\mathrm{A}}_{[8-11]})) \subset WS^{\mathrm{A}} \times WS^{\mathrm{A}}\}, \quad (8)$$

where $WS^{\mathrm{A}}_{[2-6]}$ (for all pieces) cannot be performed at the same time

$$C_5 = \{(j \in [\mathrm{st}(WS^{\mathrm{A}}_{[w]}), \ldots, \mathrm{fn}(WS^{\mathrm{A}}_{[w]})] \neq j \in [\mathrm{st}(WS^{\mathrm{A}}_{[w']}), \ldots, \mathrm{fn}(WS^{\mathrm{A}}_{[w']})]) \subset WS^{\mathrm{A}} \times WS^{\mathrm{A}}; w, w' = 2,3,4,5,6; w \neq w'\} \quad (9)$$

and also $WS^{\mathrm{A}}_{[8-11]}$

$$C_6 = \{(j \in [\mathrm{st}(WS^{\mathrm{A}}_{[w]}), \ldots, \mathrm{fn}(WS^{\mathrm{A}}_{[w]})] \neq j \in [\mathrm{st}(WS^{\mathrm{A}}_{[w']}), \ldots, \mathrm{fn}(WS^{\mathrm{A}}_{[w']})]) \subset WS^{\mathrm{A}} \times WS^{\mathrm{A}}; w, w' = 8,9,10,11; w \neq w'\}. \quad (10)$$

Priority of production can be expressed by

$$C_7 = \{(m \in P^m_n > m \in P^m_{n'} | \mathrm{st}(P^m_n(WS^{\mathrm{A}}_j)) > \mathrm{st}(P^m_{n'}(WS^{\mathrm{A}}_j))) \subset P^m_n \times P^m_n, n \neq n'\}. \quad (11)$$

As soon as the variable, the domains of values and constraints are defined, a propagation approach can be started. The aim is to restrict the values of variables (or to find such values of variables) that will satisfy all constraints. This propagation can be represented in the

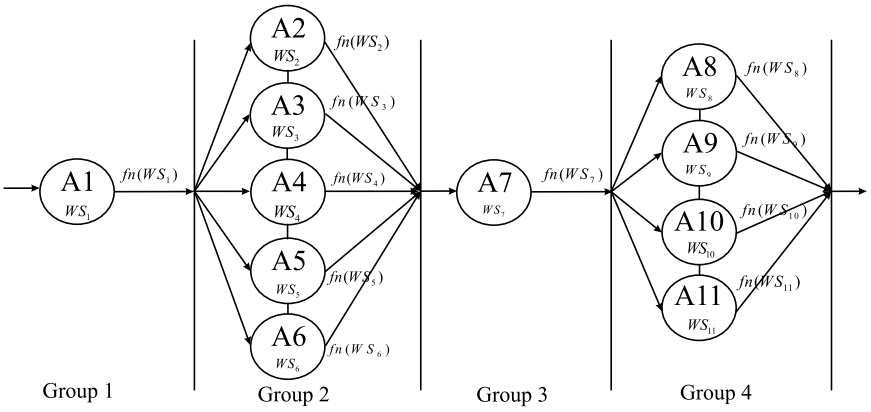

Fig. 3. Constraint network for the assignment problem.

way shown in Fig. 3. All working steps, that belong to the same workpiece, build a sequence. Every node in this sequence gets a ''finish''-position of a working step from the previous node. Using this value, a current node looks for ''start''-positions of the next working step that satisfy local constraints, calculates ''finish''-positions and propagates them to the next node. If no position satisfying the local constraints can be found, the node requests another ''finish''-position from the previous node. Thus, the network can determine locally consistent positions of all working steps. After that, the obtained values should be tested for a global consistence.

3. Application of autonomous agents

The CS (as well as CO) approach, described in the previous section, is necessary and sufficient in solving the discussed kind of assignment problem. Being implemented by one of the programming techniques, it will generate the required plan. However, working in a presence of disturbances (like machine failure, technological change and so forth) requires additional efforts to adapt the planning approach to these changes. The principles of such an adaptation are not contained in the plan itself, an additional mechanism is needed. As mentioned in the introduction, the multi-agent concept can be used as such a mechanism. This concept lends more flexibility to a manufacturing system in order to adapt it to disturbances. But what is the cause and cost of this additional feature? There is a long discussion of this point based e.g. on the decentralization (e.g. [17]) or several dynamics properties of MAS (e.g in [5]). We would like to add the following argument into this discussion.

The multi-agent system can be considered from a viewpoint of the theory of finite-state automata [18]. Transition of *m*-states automaton (with or without memory, it does not change the matter) from one state to another is determined by some rules (by a program), therefore the automaton behaves in a completely deterministic way. If a control cycle is closed (see e.g. [5]) the automaton is autonomous, i.e. behaves independently of environment (other automata). Now consider a few such automata coupled into a system *in the way that keeps their autonomy*. Forasmuch as each automaton behaves according to its own rules, there is no central program that determines a state transition for the whole system. In the ''worst case'' coupling *n* automatons with *m* states, the coupled system can demonstrate m^n states.

Evidently this ''worst case'' has never to arise in the system, but how should the behaviour of the distributed system be controlled without a central program (without a centralized mediator)? The point is that all automata are continuously communicating in order to synchronize their own states in regard to environment, to the solving task, etc. (in this case, the notion of an automaton is replaced by the notion of an agent). The agents during communication ''consider'' all possible states and then ''choose'' such a state that is the most suitable to a current problem to be solved. This is the main difference to the ''centralized programming''

approach. With ‘‘centralized programming’’ the system can only react in such a way that was preprogrammed. For example 10 agents with 10 states can demonstrate 10^{10} different combinations. However, no programmer is able to predict all situations to use all these states. Thus, if the ‘‘centralized programming’’ approach is applied to a multi-agent system, it simply restricts its behaviour, although there are essentially more abilities to react.

The sufficient number of degrees of freedom represents a key problem of the multi-agent technology. *On the one hand*, *if the system is hard restricted in the behaviour*, *such advantages of MAS as flexibility*, *emergent behaviour*, *self-organization and so on are lost. On the other hand*, *if the system has too many degrees of freedom it can communicate an infinitely long time without results*. In other words, in specific conditions only several combinations of agents states have a sense and the point is of how to achieve and to manage these states. This is a hard problem arising in many branches of science and engineering and correspondingly there are several ways to solve it. The solution suggested here is based on a hierarchic architecture of roles and emergencies that supports agent's autonomy.

Before starting to describe an approach, one methodological point concerning decentralization of multi-agent system needs to be mentioned, as shown in Fig. 4. The MAS solves a problem by using some methodological basis. For example, the CSP and COP approaches basically underlie the solution of constraint problems. The point is that a methodological basis, in almost all cases, is formulated in a centralized way. It looks like a ‘‘battle plan’’, where all agents and their interactions are shown. Therefore this global description is often denoted an *interaction pattern*.

However, the agents do not possess such a global point of view and the interaction pattern has to be distributed among agents. This decentralization concerns global information, message transfer, synchronization, decision making and so forth. The decentralized description of the chosen method should determine an individual activity of an agent as well as its interaction with other agents. It is also important that all agents behave in the ordered way, i.e. this distributed description has to include cooperation mechanisms (protocols). In order to enable a transition from the interaction pattern to the *cooperation protocol*

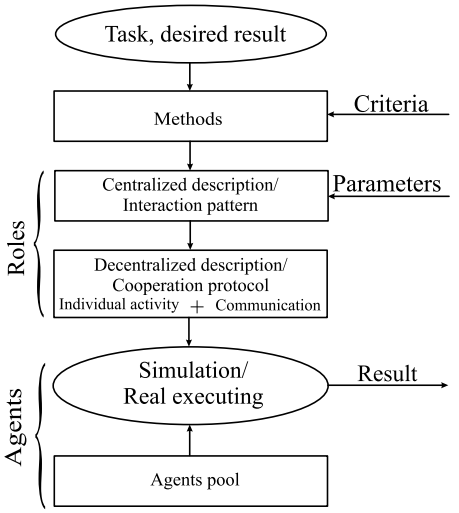

Fig. 4. Methodological approach towards agent-based applications.

(see Fig. 4) a notion of a role is introduced [19]. A role is associated with a specific activity needed to be performed (according to a methodological basis). An agent can ‘‘play’’ one role or a sequence of roles. In this way, interactions are primarily determined between roles and an agent (with corresponding abilities) handles according to the role being played at the moment. An advantage of this approach is that the centralized description (familiar for human thinking) is preserved, whereas the roles in the interaction pattern are ‘‘in fact’’ already distributed, i.e. a mapping ‘‘agent-on-role’’ may be performed in a formalized way by a program. Thus, an interaction pattern is a ‘‘mosaic image’’ that from afar looks like a common picture (method), but at a short distance is a set of separate fragments (roles). Moreover, a concept of roles allows decoupling the structure of cooperation processes from agent organizations, so that any modification of agents does not affect the cooperation process and vice versa [19].

The interaction pattern determines a *primary activity* (primary algorithm) of the multi-agent system. The primary algorithm also includes some parameters, whose modifications can be commonly associated with several disturbances (expected disturbances). A variation of these parameters does not disturb the activity of agents. A reaction of the system on the expected disturbances is incorporated into the primary algorithm. However, due to specific disturbances,

every agent can reach such a state that is not described by the primary algorithm and where performing the next step is not possible. In this case, the agent switches to the *local emergency state* and tries to resolve the arisen situation alone or with the assistance of neighbouring agents (*secondary activity*). If the abilities of an agent are not sufficient or it requires additional resources it calls a rescue agent. The rescue agent is an agent that possesses specific (usually hardware) abilities. Anyway, the aim of agents in the local emergency state is to change a part of the primary algorithm so that to adapt it to disturbances. The disturbances, causing local emergency, are predicted, but their handling is not introduced into the primary algorithm.

The primary algorithm as well as its parametrisation is optimal only for specific conditions (see Section 3.3) (e.g. combinatorial/heuristic methods for solutions of combinatorial problems, CSP/COP for constraints, etc.). If disturbances change these conditions, the primary algorithm may become non-optimal and it has no sense to repair it. All agents have to collectively recognize such a global change and to make a collective decision towards replacement of the primary algorithm. This change corresponds to a global emergency. The disturbances causing the global emergency are not expected (predicted), however they influence the conditions of primary algorithms and in this way can be recognized. Finally, there are such disturbances that cannot be absorbed by any changing of an algorithm, they remain irresolvable.

The main advantage of the concept of roles and emergencies is that an interaction pattern, defined macroscopically, can involve a huge number of roles (even more than it is required for executing primary plan), that are already *consistent with one another*. The detailed, microscopic description of every role in the cooperation protocol (i.e. a program for an agent) is generated automatically. Depending on its conditions and abilities, an agent *can execute every role in the interaction pattern*. In this way, *the MAS's degrees of freedom become bounded, but not restricted* to only one type of activity. If this number of degrees of freedom is insufficient for an adaptation of the planning approach, the system stepwisely increases it in different emergency states, until the disturbance will be absorbed or stated as irresolvable under existing conditions. This process gives the controllable flexibility to the whole MAS planning system. Now we will describe the mentioned primary algorithm as well as emergency states in the language of cooperative processes.

3.1. The primary algorithms

In the case of an assignment planning, the primary algorithm is determined by the CS approach described in Section 2 (generally usage of constraint-based approaches for MAS is not new, see e.g. [20]). Each working step in the approach is represented by a node in the constraint network, shown in Fig. 3. These nodes are separated from one another, moreover their behaviour is determined by propagations. Therefore it is natural to give a separate role to each node. However, before starting a propagation, this network has to be created and parameterized by technology, machines, number of workpieces and so on. These two steps (parameterization and propagation) will be described by interaction patterns using corresponding roles.

A role $r \in R$ (R is a set of all roles) is described by a triple $r = (D_r, \underline{z}, a)$, where $D_r \subseteq D$ is a set of services used by a role, $\underline{z}$ is a state vector, and a is an activity, that by D_r changes $\underline{z}$. Services can be understood in a sense of specific abilities that an agent has to possess in order to play a role. These abilities are connected with hardware or middleware, therefore not each agent can have them. For example, in the presented planning approach two kinds of services are required, *object-service* based on resources of operation system and *planning-service* based on CORBA resources. ''Object-service'' has primitives ''initialize'', ''find'' and has the aim to manipulate other objects. ''Planning-service'' has primitives ''*send*'', ''*receive*'' ''*calculate*'' and is used for the propagation approach. Evidently, an activity, described by a role, is based on the required services. Moreover, there are so-called *resource-objects* that possess technological information, number of processing machines, etc., i.e. they have specific resources. These ''resource-objects'' can be represented by the corresponding agents as well as by intelligent databases or a human-computer interface.

A role marking is a triple $p = (r, \mathrm{ag}, v)$, where $r \in R$, $\mathrm{ag} \in A$ is an agent, $v : D_r \rightarrow \mathrm{ag}$ operator that assigns an activity of a role to the ability of an agent.

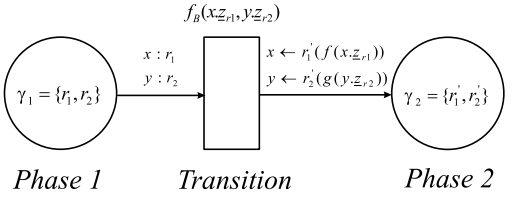

Fig. 5. Example of two phases with transition and corresponding roles.

This operator is used by matching between an available agent in pool and a role that is needed to be played. An interaction pattern describes how to solve a problem by means of interacting roles. This interaction consists of a sequence of phases. A phase $p \in P$ is a step of interactions with the set of roles γ_p in this phase. For example, let there be two phases with $\gamma_1 = \{r_1, r_2\}$ and $\gamma_2 = \{r_1', r_2'\}$ that are described by the marking $M(1) = \{(D_{r_1}, \underline{z}_{r_1}, a_{r_1}), \mathrm{ag}_1, v_1), (D_{r_2}, \underline{z}_{r_2}, a_{r_2}), \mathrm{ag}_2, v_2)\}$, shown in Fig. 5.

In the first phase two agents are playing these roles, where the abilities of the agent and activities of roles are ordered by v_1 and v_2. For a transition we define the variable x of the type *role* r_1 and the variable y of the type *role* r_2. Then, the transition is defined by the boolean function $f_B = (x \cdot \underline{z}_{r_1}, y \cdot \underline{z}_{r_2})$. After the transition, the x-connected mark loses the role r_1 and gets the role r_1', where the information transfer from r_1 to r_1' is defined by f. The same applies also to the y-connected mark, where g describes the corresponding information transfer. After the transition we have $M(2) = \{(D_{r_1}, f(\underline{z}_{r_1}), a_{r_1'}), \mathrm{ag}_1, v_{1,r_1'}), (D_{r_2'}, g(\underline{z}_{r_2}), a_{r_2'}), \mathrm{ag}_{2,r_2'}, v_{2,r_2'})\}$. In this case, it is a priori known, the role r_1' requires the same services as r_1, therefore ag_1 can play it further.

An interaction network is defined by the following tuple $PN = (P, T, F, \sigma_i, \sigma_t, \sigma_o, M_0)$, where P is a set of places, T is a set of transitions, $P \cup T \neq 0$, $P \cap T = 0$, $F \subseteq (P \times T) \cup (P \times T)$ is the flow-relation, $\sigma_i : (P \times P) \rightarrow N$ is an input function, defining the directed arc from places to transitions, $\sigma_o : (P \times P) \rightarrow N$ is an output function, defining the directed arc from transitions to places, $\sigma_t : T \rightarrow N$ is the guard of transitions, M_0 is the initial marking. In this way, the interaction pattern is described by the modified Petri net, details can be found in [19].

As already mentioned, the primary algorithm consists of two parts, parameterization and propagation, that represent a linear sequence of activities. The parameterization part, shown in Fig. 6, has three phases p_0, p_1, p_2 whose main result consists of determining a structure, neighbourhood relations and parameterization of nodes of the constraint network. The roles γ_0, γ_1 are "Initializers" of WS-order and WS-nodes correspondingly. The role γ_0 is activated by the first production order. This role reads resource-objects and determines how many nodes (WS-roles) are required. The transition t_0 proves, whether the result of *j.returnEND*() is true (action is successful) and activates γ_1 with the parameter n as the number of required nodes. The γ_1 initializes each node according to all restrictions (technology, propagation rules, number of machines and so on). If this activity is also successful (transition t_1), the third role γ_2 is activated. It connects the created nodes (return a pointer to previous node), composing in this way a network. This interaction plan is finished (transition t_3) if no further nodes exist that are needed to be connected.

The propagation part, shown in Fig. 7, consists of three blocks: local (the phases p_3, p_4, p_5) and global (the phase p_6) propagations and an activity (the phase p_7) in the case of empty sets. The roles γ_3, γ_5 determine the propagation in the first and the last nodes, whereas γ_4 does the same for all other nodes. The transition t_7 verifies, whether the local propagation was successful for all nodes and activates the global propagation in γ_6. We emphasize that the local propagation requires a sequential execution of the roles, whereas in global propagation all roles can be executed in parallel.

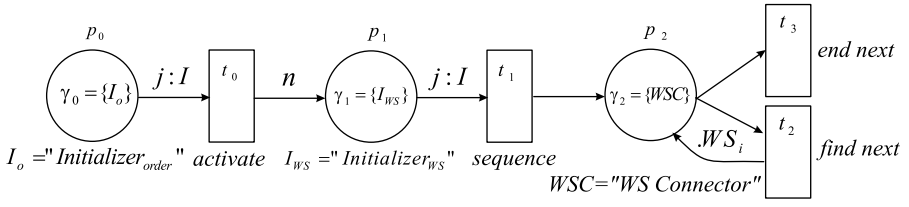

Fig. 6. Primary algorithm: parameterization part.

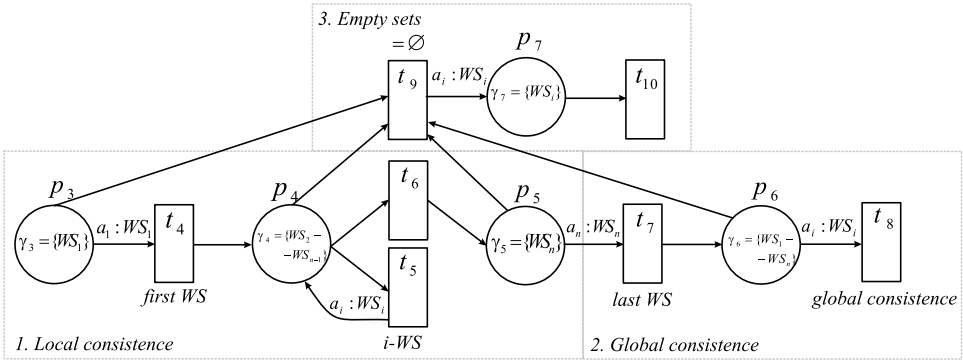

Fig. 7. Primary algorithm: propagation part.

Finally, the transition t_9 proves, whether the values set (WS-positions) of each node is empty. In the case of empty sets the role γ_7 tries to increase initial areas of values, first locally in neighbour nodes, then globally by the restart of the local propagation. Descriptions of phases and transitions from Figs. 6 and 7 are collected in Tables 3 and 4.

Considering these interaction patterns, we see they include several parameters (e.g. number of available machines, time slot, technology, etc.), that can appear as expected disturbances. Moreover, these patterns can hierarchically include such activities, that are not directly connected with a CSP problem, e.g. control and optimization of logistical operations. Consequently the MAS, even in the phase of primary activity, has enough degrees of freedom to behave adaptively. As followed from the experiments with the planning system, approximately 90% of all one-step, short-term disturbances and 30–40% one-step, middle-term disturbances can be absorbed in the primary phase.

3.2. The local emergency and rescue agents

As mentioned, an agent playing a role performs some activities, whose results are tested by the transition. The local emergency arises, when the results at transitions are of such a type, that cannot be processed by the transition or there are generally no results. In both cases an agent cannot finish a current role and execution of a common interaction pattern is stopped. The local emergency has a natural analogy in real manufacturing. The manufacturing can be disordered by failures, by absence of resources, by fire and so forth. In each case the reason of disorder is different, however, the classification enables to react on a disorder in a predicted way. In each case there is a schema of how to react, e.g. at fire alarm. The interaction patterns of local emergency are similar to these schemes, where specific resources/abilities, like a fire brigade, are represented by rescue agents.

The agent, playing a current role, cannot perform recognition of an emergency. For this aim a so-called

Table 3
Description of phases in cooperation patterns shown in Figs. 6 and 7

Phase	Roles
$p_0, \gamma_0 = \{(I)Initializer_{order}\}$	$I = (D_r = \{services\}, \underline{z} = (objects), act =< initialize.object, technology.getNumberOfWS >)$
$p_1, \gamma_1 = \{(I)Initializer_{WS}\}$	$I = (D_r = \{services\}, \underline{z} = (objects), act =< initialize.WS_i >)$
$p_2, \gamma_2 = \{(WSC)WS - Connector\}$	$WSC = (D_r = \{services\}, \underline{z} = (objects), act =< find.next(a_i) >)$
$p_3, \gamma_3 = \{WS_1\}$	$WS_1 = (D_r = \{planning\}, \underline{z} = (positions), act =< WS_1.sendV, WS_1.calculateL >)$
$p_4, \gamma_4 = \{WS_i\}$	$WS_i = (D_r = \{planning\}, \underline{z} = (positions), act =< WS_i.sendV, WS_i.calculateL >)$
$p_5, \gamma_5 = \{WS_n\}$	$WS_n = (D_r = \{planning\}, \underline{z} = (positions), act =< WS_n.sendV, WS_n.sendG, WS_n.calculateL >)$
$p_6, \gamma_6 = \{WS_1, \ldots, WS_n\}$	$WS_i = (D_r = \{planning\}, \underline{z} = (positions), act =< WS_i.calculateG >)$
$p_7, \gamma_7 = \{WS_i\}$	$WS_i = (D_r = \{planning, services\}, \underline{z} = (positions), act =< WS_i.send >)$

Table 4
Description of transitions in cooperation patterns shown in Figs. 6 and 7

T	Description
t_0	$\sigma(p_0,t_0)=(j:I)$, $\sigma(t_0)=j.returnEND$, $\sigma(t_0,p_1)=n\leftarrow j.numberOfWS$
t_1	$\sigma(p_1,t_1)=(j:I,\forall a_i:WS)$, $\sigma(t_1,p_2)=a_i\leftarrow pointer(WS_i)$, $\sigma(t_1)=j.returnEND(1)\&\ldots\&j.returnEND(n)$
t_2	$\sigma(p_2,t_2)=(j:WSC,\forall a_i:WS)$, $\sigma(t_2)=j.returnNext$ $\sigma(t_2,p_3)=\{i++,a_i\leftarrow pointer(a_{i-1})\}$
t_3	$\sigma(p_3,t_3)=(j:WSC)$, $\sigma(t_3)=!j.returnNext$
t_4	$\sigma(p_3,t_4)=(\forall a_i:WS)$, $\sigma(t_4)=a_1.retValues\neq\emptyset$, $\sigma(t_4,p_4)=\lambda\leftarrow a_1.retValues$
t_5	$\sigma(p_4,t_5)=(\forall a_i:WS,\lambda:Positions)$, $\sigma(t_5)=a_i.retValues\neq\emptyset\&i<n$, $\sigma(t_5,p_4)=\{\lambda\leftarrow a_i.retValues,i++\}$
t_6	$\sigma(p_4,t_6)=(\forall a_i:WS,\lambda:Positions)$, $\sigma(t_6)=a_n.retValues\neq\emptyset$, $\sigma(t_6,p_5)=\lambda\leftarrow a_i.retValues$
t_7	$\sigma(p_5,t_7)=(\forall a_i:WS)$, $\sigma(t_7)=a_n.retValues\neq\emptyset$, $\sigma(t_7,p_6)=\forall a_i\leftarrow$ "FinishLocal"
t_8	$\sigma(p_6,t_8)=(\forall a_i:WS)$, $\sigma(t_8)=a_i.retValues\neq\emptyset$
t_9	$\sigma(\{p_3,p4,p5,p6\},t_9)=(\forall a_i:WS)$, $\sigma(t_9)=a_i=\emptyset$, $\sigma(t_9,p_7)=i\leftarrow a_i$
t_{10}	$\sigma(p_7,t_{10})=(\forall a_i:WS)$, $\sigma(t_{10})=a_i.returnEND$

activity guard agent is needed, whose macroscopic interaction pattern is shown in Fig. 8. In phase p_0 it observes an execution of agent's activities and in the case either the "wrong type of variables" or "time out" messages at transition occur the guard agent activates the phase p_1. Here, the concept of the error-return-code (ERC) is utilized, that allows identification of the arisen problem. A typical example of the basic ERC is I/O messages known in each programming language. Each activity, performed by an agent, is equipped with a set of ERCs, including resource-ERC, activity-ERC and so forth. Based on this returned code, a specific problem solving process can be started. If an ERC is returned, the phase p_2 is activated, otherwise the software-rescue agent (role γ_3) is called. The problem solving process in phase p_2 requires specific resources and abilities. If a current agent possesses these, this agent (roles γ_2,γ_4) resolves the problem. Otherwise a specialized rescue agent (role γ_5) with the required abilities will be called. The roles γ_2,γ_4 represent in turn the interaction patterns of a complex nature that are hierarchically called when the corresponding role is activated. The problem types used for the assignment of rescue agents are summarized in Table 5. The interaction pattern for the problem-oriented rescue roles is not described, because their reaction is evident and moreover they depend on the implementation details.

However, in the case of a "time out" error, an agent cannot form an error-return-code. This situation points to the case, when an agent cannot accomplish an internal activity cycle not because of the absence of resources (i.e. problem-oriented emergency), but because of internal confusion in an agent's program. In this case, the software-rescue (SR) agent should be called. The first aim of the SR-agent is to prove an internal structure of an agent in relation to current data from sensors and communications. This can point to internal software errors. The second aim is to ascertain the problems in external activity that lead to the "time out" emergency.

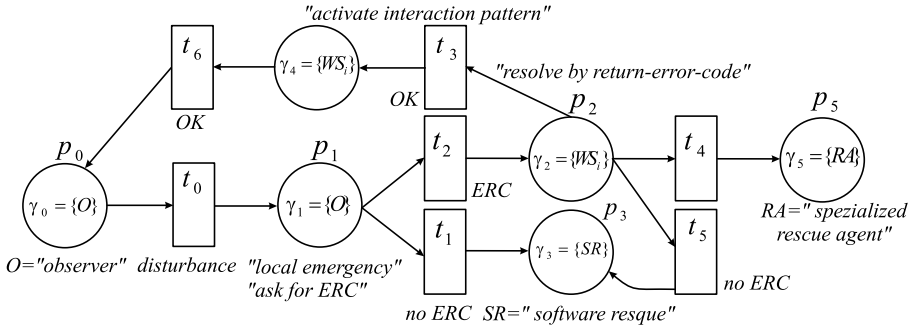

Fig. 8. Macroscopic interaction plan for local emergency.

Table 5
Types of specialized rescue agents

Type of specialized rescue agent (RA)	Type of problem	Input parameter	Type of reaction
Input resource RA	Absent	Type of resources	To supply
Output resource RA	Absent	Type of resources	To supply
Object-availability RA	Not available	Object	To provide
Technology RA	Not executable	Working step, object	To replace
Software RA	Not responded	Agent	Simulation

The idea of the SR agent is based on the autonomy cycle, shown in Fig. 9. The autonomy cycle represents the common steps that any agent cyclically executes. If there is an emergency of this kind, it means a problem has arisen on some of these steps. The SR agent simulates the distorted role (roles) first with ideal input/output data. When this simulation is successful, the SR-agent replaces stepwisely each input/output with real ones, until the problem is brought to light. However, there exists one critical point concerning the reaction of the SR-agent on the detected problem. Generally, any problem on this step means, that the agent does not correspond to the environment. The adequate reaction of the SR-agent can include activity to modify the agent or its environment. Modification of the environment can be performed either by a specific rescue agent or simply by sending a message to operator, however, a modification of the internal program structure of an agent causes several problems. This step includes developing a program generator that will be briefly discussed in the next section. In the given implementation, this problem still remains unresolved and the SR-agent reacts on the problem only by sending a corresponding error-message.

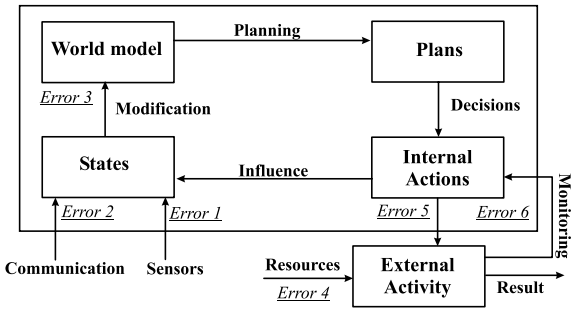

Fig. 9. Autonomy cycle of an agent. "Error 1": wrong type of sensor data, "error 2": this type of communication data is not expected, "error 3": world model does not support the input data, "error 4": wrong type of resources, "error 5": external activity is not responded, "error 6": wrong type of external activity.

3.3. The global emergency

A global emergency arises, when the multi-agent system is no longer able to follow the primary algorithm and to react adequately on the emerged disturbances. The disturbances causing the global emergency cannot be nearly identified, however, they can be recognized by the effects left over. Firstly, they disturb the global criteria underlying methodological approach so that it is no longer valid or, secondly, a local emergency is not resolvable even by rescue agents.

In the first case, the disturbances achieve some qualitative threshold that completely disturbs the primary algorithm. For example, if the technology will be changed so that most of the restrictions will disappear. This leads to an agent's state space, such that after the CS approach is equal (or almost equal) to initial space, i.e. the original methodological assumption is no longer valid. The mentioned disturbances have a global nature that influences all agents. In order to recognize this effect, all agents have to perform a negotiation and to make a collective decision [21].

The second reason, causing global emergency, is an irresolvable local emergency. Before declaring a state to be generally non-resolvable, an agent (even a rescue agent) transfers information about the problem to other agents in hoping they have the needed resources/abilities and can solve the problem. This solution may have also a local or global form. In the local case another agent takes over the solution of the problem that is equivalent to the local emergency discussed in the previous section. The global form means the mentioned global change.

Table 6
Problems causing global emergency and their resolution

Type of problem	Consequence	Resolving
Number of technolocal restrictions is highly reduced	State spase overflow, CS approach fails	Use combinatorial optimization
Frequent disturbances, especially in resources	Frequent replanning, irregular filling of production's queue	Send corresponding message
Frequently unavailable communication	Propagation fails	Restart of network components

Now, the point is of what kind of global change is required by a global emergency in the agent group? If the methodological approach is no longer valid or the agents, using all current interaction pattern, are not able to resolve the arisen problem, it is natural to change this interaction pattern. However the question is which new interaction pattern should arise (and of no lesser importance—how should it arise)? Generally, there are two possibilities, either to use pre-formulated interaction patterns or to generate this pattern dynamically by request. The generation of an interaction pattern represents a separate topic involving software-specific questions (e.g. language semantics, parsers and so on) and treatment of more general algorithms of tasks decomposition. One such algorithm, so-called algorithm of symbolic tasks decomposition (ASTD) based on some synergetic properties of collective systems, has been already developed and is being tested for manufacturing-specific problems [22]. Treatment of such a generator, because of the complexity of this issue, represents the focus of a separate work.

However, several cases of global emergency in the planning approach can be covered by the reserve interaction patterns (see Table 6). If the measures undertaken in the case of a global emergency are insufficient and do not lead to resolution of the problem, the multi-agent system declares the current state as irresolvable.

4. Agent-based optimization

The steps, described in the previous sections, allow generating the sequence of working steps that satisfies all local constraints. However such global characteristics of a plan as cost, manufacturing time and so forth are not considered. Therefore, as pointed out by some authors, the next step consists of optimizing the obtained sequences. Generally speaking, constraint satisfaction and optimization cannot be separated into two different steps, rather, it represents a sui generis combination. Before starting a discussion of agent-based optimization, two features of such an approach need to be mentioned.

The first feature of agent-based optimization consists in a local character of used data. Combinatorial or heuristic approaches assume the data, required to be optimized, are globally available. Optimization in this case looks like a ''chess game'', where all the pieces are visible and some combination of piece positions is needed to be found. In the multi-agent variation there is no this central viewpoint, each agent makes only a local decision about to occupy a positions or not (see Fig. 10). In this way, the agent performs *optimization of local decisions instead of global positions of the working steps*. Moreover, from the agents viewpoint, *any of their decisions has no foreseeable perspectives for a global optimum.*

The second feature of agent-based optimization is caused by the local nature of the optimization problem. Each agent during the CS phase tries to occupy a position immediately after the previous working step. This strategy is motivated from the manufacturing side in trying to avoid a waiting time at processing elements (machines). Evidently, this strategy cannot guarantee a global optimum. Therefore the agent has to compute what will happen if the next processing step will not begin immediately after the previous step. It can be achieved by shifting a manufacturing of a workpiece on some steps, that increases a local cost of a plan (e.g. intermediate storage), but reduces global costs (see Fig. 11). This approach (forecasting) is similar to a decision tree in distributed form. As known, an increase of the depth of tree rapidly increases the search space.

After discussing the features of agents-based optimization, one can focus on the problem of assignment planning. There are two important steps, that the optimization needs to be performed on. Firstly, an order of the working steps in the groups 2 and 4 (see Fig. 3). Forasmuch as there are only 2881 combinations between WS_1–WS_{11}, this optimization step can be performed by exhaustive search. The second point of

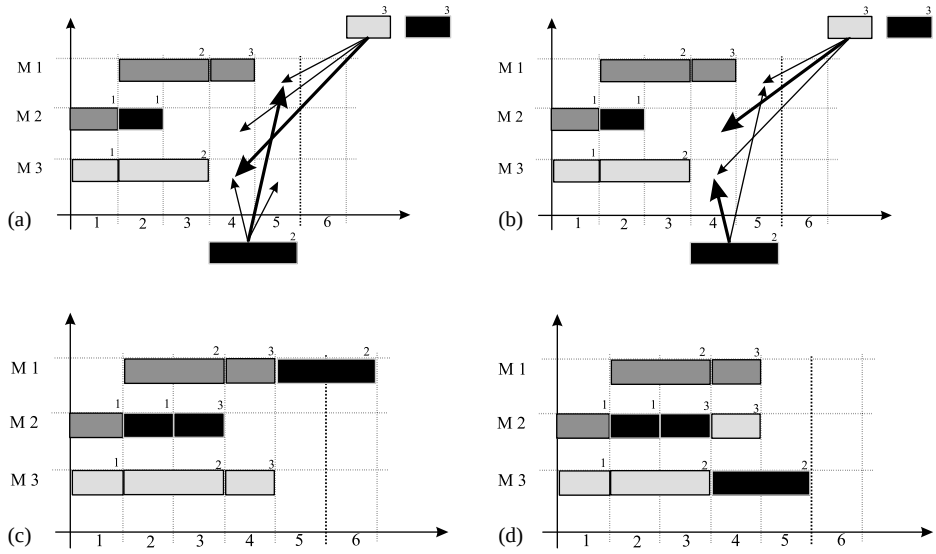

Fig. 10. Example of the plan making for the minimal time and minimal transportation cost. There are three pieces to be manufactured (P_1 grey, P_2 white, P_3 black). Processing of each piece consists of three working steps (with length 1, 2, 1 correspondingly). The working step 2 cannot be processed on the second machine M2. Each working step is represented by an agent. (a) Decisions of the agents, where $WS_2(P_3)$ is going to M1 in position 5, $WS_3(P_2)$ is going to M3 in position 4. Local decision is based on the criterion of minimal transport cost; (b) decisions of the agents, where $WS_2(P_3)$ is going to M3 in position 4, $WS_3(P_2)$ is going to M2 in position 4. Local decision is based on the criterion of minimal length of this plan; (c) final plan corresponding to (a) and (d) final plan corresponding to (b).

optimization concerns the local decisions (concerning machine and position) made by agents. However, the search space (taking into account the forecasting effect) grows in this case exponentially and e.g. even for 22 agent (2 production workpieces, forecast for next five positions) comes to $\sim 10^{10}$. Therefore exhaustive search methods like constraints optimization are inefficient even on a very fast computer. The search space can be essentially reduced if the following observation is taken into account.

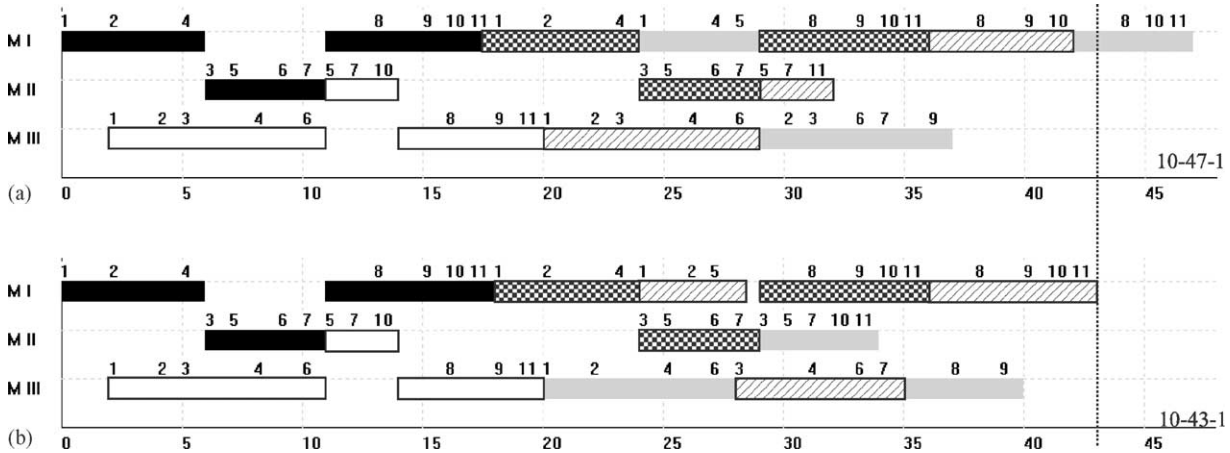

Fig. 11. The "forecasting" effect in assignment planning. (a) Each working step begins immediately after the previous step. The length of a whole plan is equal to 47 and (b) the start of the working step 8 (third piece) is delayed on one step, that allows reducing the common length to 43 steps.

The assignment planning for different workpieces represents an iterative process, where all iterations are very similar to one another. In this way, the whole assignment plan represents a periodic pattern, that can be observed in Fig. 11. Here there are two main patterns, shown by black and white colors (order of the working steps as well as their positions on machines) that, however, differ in the last workpieces. It means that in the case where the optimal (or near optimal) scheme for the first iteration is found, the next iteration can use the same scheme. The distributed approach being able to treat this kind of pattern-like problem is known as the ant colony optimization algorithm (ACO) [23]. This method originated from the observation of ants in their colony. Going from the nest to the food source, every ant deposits a chemical substance, called pheromone, on the ground. In the decision point (intersection between branches of a path) ants make a local decision, based on the amount of pheromone, where a higher concentration means a shorter path. The result of this strategy represents a pattern of routes, where thick line points to a shorter path. A similar strategy can be applied to the local decisions of agents, participating in the plan making.

Agents, after the CS approach, choose several assignment plans from the generated set and form an optimization pool. These assignment plans can also represent simple segments of plans (these connected working steps represent independent parts of the assignment plan) that satisfy all formulated constraints. These segments/plans can be combined into a common plan, so that to satisfy the postulated optimization criterion. Thus, the more optimal segments that are included into this pool, a more optimal common plan will be obtained. The ACO algorithm marks (like a pheromone rate) the optimal segments obtained on the previous step. The fragments with the highest pheromone rate are included into the top of the pool. In this way, agents consider first the ACO-obtained sequence and try to modify it (e.g. using forecasting effect). Thus, an optimization pool always contains solutions with a high pheromone rate, from them the most optimal one will then be chosen.

The optimality of a plan is also influenced by a number of transportations of a workpiece from one machine to another. These transportations are represented by so-called ‘‘jumps’’ in the plan making, as shown in Fig. 12. The minimal number of jumps for a workpiece is defined by technological requirements and e.g. for a plan shown in Fig. 11 is equal to 2. However the number of jumps can be increased so that increases the cost, but improves other characteristics of an assignment plan. This mechanism is utilized in combined optimization criteria, e.g. the minimal cost at defined length (constant delivery date). Dependence between the number of jumps and, for example, the length of generated plans is shown in Fig. 13.

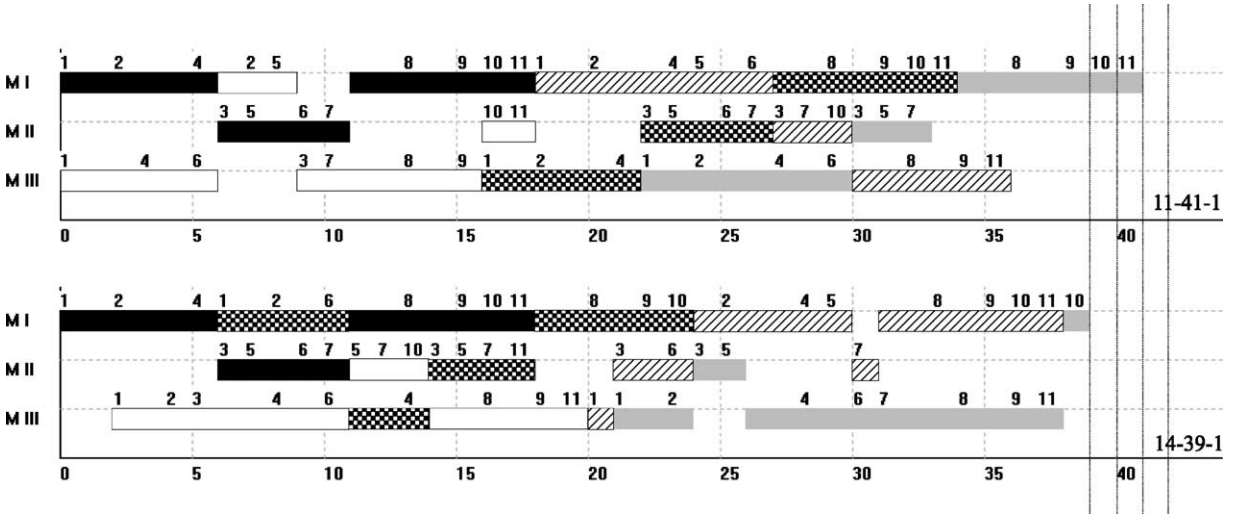

Fig. 12. Nonoptimal assignment plans with different number of jumps that result in different length and transportation costs.

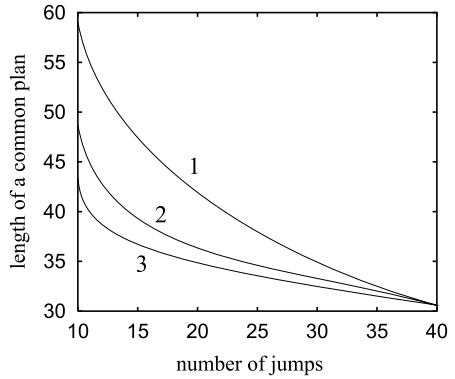

Fig. 13. Dependence between the number of jumps and the length of generated plans, where the curves 1–3 represent correspondingly the cases without forecasting, with one-position forecasting for only the first workpiece, and with two-positions forecasting for all workpieces.

The last remark concerns optimization criteria. Conventional criteria are a minimal cost and minimal manufacturing length (time). As mentioned, the represented approach can also use a manufacturing cost, transportation cost and storage cost, defined for each machine as well as each working step. The assignment plans, shown in Fig. 11, are optimized, as an example, with the criterion of a minimal transportation cost.

5. Discussion

The discussion includes two main points that concern a classification of disturbances and remarks towards planning approach. Turbulences, as suggested in [24], can be classified into four main groups: management, organizational, technological and resource ones. From the viewpoint of multi-agent system these disturbances can be expected and included into the primary algorithm. Disturbances represent in this case some external parameters that control a planning process. Next, disturbances can be predicted, but, however, not included into the primary algorithm, because they occurs seldom and have a specific nature. These disturbances generally causes the local emergency. The group of unexpected disturbances is also divided into two parts by their effect. If the effect of disturbances changes several global parameters, the system can absorb them by equivalent global changes. The rest of unexpected disturbances builds the last group of irresolvable disturbances. Correspondence between the disturbances and reaction of the multi-agent system is shown in Table 7.

In the performed simulations we have reproduced the most typical disturbances arising in real manufacturing [25], to test the MAS planning system. *These disturbances are introduced after the planning phase and immediately before manufacturing*, *as well as during the manufacturing phase*. In the last case replanning takes into account the already manufactured part of the distorted plan, and so the MAS-based planning in fact accompanies a manufacturing process. As it turned out, the system is stable to almost all one-step, short-terms disturbances (except exotic ones like ‘‘a supply is completely broken down’’). The majority of one-step, middle-term disturbances can be also successfully treated, but it depends on the abilities of the rescue agents. Generally, the more abilities that are delegated to the agents, the wider the spectrum of disturbances that can be automatically absorbed. However, the great problem arises at many-steps disturbances (several disturbances simultaneously or in a short time slot, so that they nonlinearly influence each other). We suppose the complexity of such a problem oversteps, in many cases, today’s possibilities of algorithmic problem solving. For

Table 7
Equivalence between disturbances and a reaction of multi-agent system

Disturbances on:	Example	Type of reaction	Reaction time
Management level	Aims, appointments, deadlines	Not expected, not included, expected but not included	Long-term, middle-term
Organizational level	Orders, lot size, urgency	Expected and included	Short-term
Technological level	Product-technology, process-technology	Expected but not included, not expected, not included	Short-term, long-term
Resources level	Machines, supplies	Expected and included	Short-term

the long-term disturbances the system has been not tested.

The presented approach integrates a process planning and manufacturing control in the multi-agent way. On the one hand, utilizing the concept of roles and emergency states, the complexity of multi-agent system is bounded, *this makes the problem of agent-based scheduling tractable*. On the other hand, the planning system still possesses enough degrees of freedom to react reasonably and adaptively to disturbances. Thus, the fundamental problem of a relation between complexity and flexibility, known from other MAS-approaches, is solved here in this way. The approach does not require any centralized elements. That, firstly, enables a distributed implementation, secondly, essentially increases a reliability of common system. However, such problems as a treatment of irresolvable disturbances in the global emergency state still remain unsolved and require more general algorithms of tasks decomposition, that represent an open research field in modern manufacturing as well as in computer science.

Acknowledgements

The presented work is supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) in the framework of SFB 467 (Sonderforschungsbereich 467) ''Transformable Business Structures for Multiple-Variant Series Production''.

References

[1] H.-P. Wiendahl, Wandlungsfähigkeit, wt Werkstattstechnik Jahrgang 92 (4) (2002) 122–127.

[2] P. Peeters, T. Heikkilä, S. Bussman, J. Wyns, P. Valckenaers, H. van Brussel, Novel manufacturing system requirements in automated, line-oriented discrete assembly, in: Proceedings of the 4th IMS-WG Workshop, University of Nancy, Nancy, 1998, p. 10.

[3] B.J. Pine, Mass Customization. The New Frontier in Business Competition, Harvard Business School Press, Boston, MA, 1999.

[4] S. Bussmann, K. Schild, Self-organizing manufacturing control: an industrial application of agent technology, in: Proceedinhs of the ICMAS'2000, MA, Boston, 2000, pp. 87–94.

[5] G. Weiss, Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.

[6] T. Sandholm, Negotiation Among Self-interested Computationally Limited Agents, Ph.D. Thesis, University of Massachusetts, Amherst, 1996.

[7] V. Dörrsam, Materialflussrientierte Leistungsanalyse einstufiger Produktionssysteme, Ph.D. Thesis, University of Karlsruhe, Karlsruhe, 1999.

[8] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice-Hall, 1995.

[9] S. Graves, A.R. Kan, P. Zipkin, Logistics of production and inventory, Handbooks in Operations Research and Management Science, vol. 4, North Holland, 1993.

[10] H. Haken, M. Schanz, J. Starke, Treatment of combinatorial optimization problems using selection equations with cost terms. Part I. Two-dimensional assignment problems, Physica D (134) (1999) 227–241.

[11] J. Blazewicz, W. Domschke, E. Pesch, The job shop scheduling problem: conventional and new solution techniques, European Journal of Operational Research 93 (1996) 1–33.

[12] K. Alicke, Modellierung und Optimierung von mehrstufigen Umschlagsystemen, Ph.D. Thesis, University of Karlsruhe, Karlsruhe, 1999.

[13] R. Bellman, S. Dreyfus, Applied Dynamic Programming, Princeton, NJ, 1962.

[14] E.H. Durfee, Distributed problem solving and planning, in: G. Weiss (Ed.), Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence, MIT Press, 1999, pp. 121–164.

[15] G. Pritschow, A. Storr, M. Weiner, Integrated, operator-oriented order and process planning for flexible manufacturing, Production Engineering VIII (1) (2001) 103–106.

[16] ISO/DIS14649-1, Part 1: Overview and Fundamental Principles, Genf Final DIS, 2000.

[17] H. Tönshoff, P.-O. Woelk, I. Timm, O. Herzog, Flexible process planning and production control using co-operative agent systems, in: Proceedings of the COMA'01, Stellenbosch, South Africa, 2001, pp. 442–449.

[18] J. Carroll, D. Long, Theory of Finite Automata, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[19] M. Muscholl, Interaction und Kooperation in Multiagentsystemen, Ph.D. Thesis, University of Stuttgart, Stuttgart, 2001.

[20] A. Nareyek, Constraint-based agents, Lecture Notes in Computer Science, vol. 2062, Springer-Verlag, 2001.

[21] O. Kornienko, S. Kornienko, P. Levi, Collective decision making using natural self-organization in distributed systems, in: Proceedings of the CIMCA'2001, Las Vegas, USA, 2001, pp. 460–471.

[22] S. Kornienko, O. Kornienko, P. Levi, Multi-agent Repairer of Damaged Process Plans in Manufacturing Environment, in: Proceedings of IAS-8, Amsterdam, 2004.

[23] D. Corne, M. Dorigo, F.G. (Eds.), New Ideas in Optimization, McGraw-Hill, 1999.

[24] SFB467, Report of sfb 467 on 05.12.2001, Preprint, University of Stuttgart, 2001.

[25] SFB467, Studie 'Turbulenz und Wandlungsfähigkeit', 2002–2003, Preprint, Fraunhofer IPA, IFF Universität Stuttgart, 2002/2003.

J. Priese graduated in industrial engineering and management at the University of Rostock, Germany in 2001. Current position (since 2001): research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) in Stuttgart, Germany. His main research fields are process and work planning as well as manufacturing execution systems. He was born in 1975 in Röbel, Germany.

O. Kornienko received initial educational degree (BS degree) at the Moscow State Technical University n.a. N.E. Bauman (MSTU) in 1991 and, in 1994, she graduated in computer science (DiplEng (MS)) at the Taganrog State University, Russia with honorary diploma. In 1994–1996 she did a graduate study towards candidate of science (PhD) degree in computer science and was a lecturer at this university. In 1996, she received award of the Russia's Presidential Fellowship for research abroad and the DAAD's (German Academic Exchange Service) research grant. In 1997, she received the DAAD's second research grant for scientific work in the Center of Synergetics, headed by Prof. H. Haken. Currently, she is a research assistant in the Institute of Parallel and Distributed Systems at the University of Stuttgart and is finishing her PhD thesis. Olga's main research interests include distributed AI, cooperative problem solving and decision making in multi-agent systems, applications of these technologies in robotics and manufacturing.

S. Kornienko is a lecturer at the University of Stuttgart, Germany. He graduated in computer science (DiplEng (MS)) at the Taganrog State University, Russia in 1994. After that he was for 2 years a department head of new informational technologies at this university. In 1993-1996, he has taken part in several industrial projects, such as distributed gas-measurement data acquisition system, control of parallel processes in manufacturing of semiconducting material. In 1996, he received award of the Russia's Presidential Fellowship for research abroad, and, in 1997, the DAAD's (German Academic Exchange Service) research grant for scientific work in the Center of Synergetics, headed by Prof. H. Haken. Currently, he is finishing PhD thesis in the Institute of Parallel and Distributed Systems at the University of Stuttgart. Sergey's main research interests include self-organization and synergetic phenomena in artificial collective systems (like soccer-robots or nanorobots), agent- and multi-agent technologies, as well as their applications in the field of robotics and manufacturing. He is author and co-author of over 30 papers in international journals and conferences.