Proceedings of the 21st IASTED International Conference
APPLIED INFORMATICS
February 10-13, 2003, Innsbruck, Austria

Flexible manufacturing process planning based on the multi-agent technology

S. Kornienko, O. Kornienko, P. Levi

Institute of Parallel and Distributed High-Performance Systems
University of Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany

ABSTRACT

Flexible manufacturing systems working in highly dynamic modern markets requires innovative concepts towards flexible management, production scheduling and planning. The agent-based technology provides a desired flexibility, supports a short reaction time and can underlie the required planning mechanismes. Moreover it does not require any centralized elements allowing a distributed implementation and essentially increasing a reliability of common system. A specific software architecture supports this flexibility and enables an adaptive reaction on disturbances often arising in production.

KEY WORDS
Flexible manufacturing system, multi-agent systems.

1 Introduction

Globalization of national markets created a high level of competition that only quickly reconfigurable production systems can survive in. That concerns not only the development processes, that should be essentially shortened, but also the manufacturing process itself. Manufacturing should become oriented to fabrication on demands of client with unique specifications (e.g. [1]). These requirements mean a flexibility of almost all processes of a modern factory and primarily relates to planning processes. Planning, especially the process planning, has to be distributed and stable to different changes (disturbances) corresponding to ”fabrication on demand” as well as to production malfunctions.

The concept of multi-agent system (MAS) [2] can underlie the mentioned planning processes. The MAS-based processes are distributed, moreover the ”negotiation-based programming” provides for the problem solving (decisions making, planning, etc.) essentially more degree of freedom than traditional centralized systems. That allows reacting to short-term disturbances, supporting self-maintenance, integrating into autonomous manufacturing. The agents behaves autonomously, takes decisions and can communicate with human or non-human workers. It enables creating complex autonomous manufacturing structures.

The represented further approach is addressed to the agent-based process planning, where the production orders should be assigned to available machines, taking into account the technological, organizational restrictions as well as optimization criteria. The main point of consideration is focused on a flexible structure of multi-agent system that is provided by specific software architecture. This structure allows the planning system to adapt to changes of environment so that to perform the planning successfully further. The main notions of the suggested approach are described in form of specification language, based on the extended Petri networks. This exemplifies the proposed ideas and shows several implementation details. Finally, the agent-based optimization is briefly discussed.

2 Process planning problem

The following scenario is based on a flexible manufacturing system (FMS). The FMSs represent a sequence of the processing machines, being able to perform different manufacturing operations without retooling. In the given example the FMS has to manufacture a multitude of parts and variants of workpieces determined by corresponding production orders. A manufacturing of a workpiece consists of different steps and requires a corresponding plan that is known as process planning. Moreover each type of workpieces has own technology, i.e. the manufacturing consists of different working steps (WS). These WS are defined by a technological process, generally all these WS cannot be processed on one machine. Each from the working steps has different length and cost.

The working steps can be executed only by the specific machines that are able to perform the required operations. Moreover there is a technological order of these operations (e.g. milling only after boring). Each working step needs a preparation time that is added to common length of working step. The general 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 mentioned restrictions.

As can see, this problem belongs to so-called ”constraints” 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 constraints satisfaction (CS) problem and constraints optimization (CO) problem. In the given work this methodological way is used.

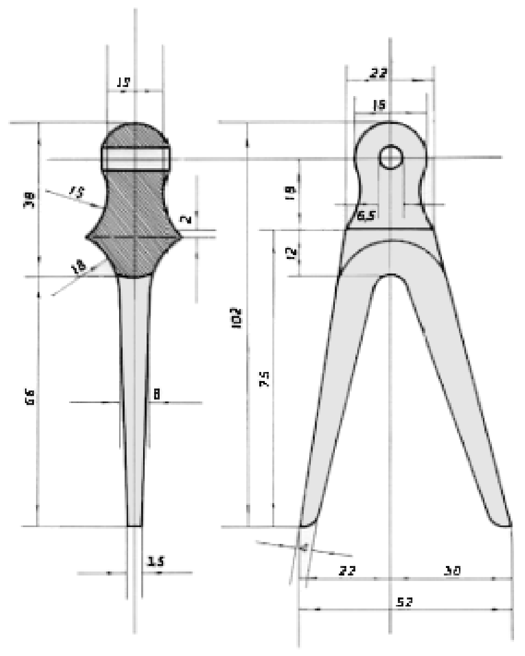

Figure 1. *Example of a workpiece.*

A processing of the workpiece shown in Fig. 1 consists (as an example) of eight WS that include such operations as boring, milling, grinding in a defined order. These working steps can be represented in the form shown in Fig. 2, that is usually denoted as constraint network. Each node is a variable with the corresponding domain of values, representing the positions of a working step in a manufacturing queue. The aim of the approach is to restrict the values of variables (or to find such values of variables) that will satisfy all constraints.

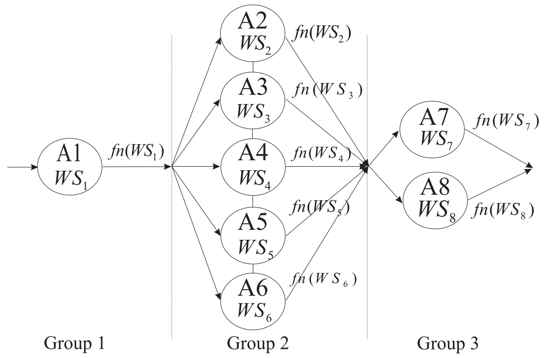

Figure 2. *Constraint network for the workpiece in Fig. 1.*

Every node in this network gets a ”finish”-position of a working step from 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 local constraint can be found, the node requests another ”finish”-position from previous node. In this way the network can determine locally consistent positions of all WS. After that the obtained values should be tested for a global consistence. In this way the CS approach defines the consistent positions (an assignment plan) of all WS in queues. However this assignment plan is not optimal and requires the further optimization.

As mentioned, the main problem of process planning consists in often arisen disturbances (like machine malfunctions or urgent production orders). Several types of these disturbances is shown in Table 1. Therefore the goal

Disturbances on:	Example	Reaction
management level	aims, appointments, deadlines	long-term middle-term
organizational level	orders, lot size, urgent order	short-term
technological level	product-technology process-technology	middle-term long-term
resources level	machines, supplies	short-term

Table 1. *Several types of disturbances.*

of planning approach is not only to generate in distributed way an optimized plan satisfying all restriction but also to do it in presence of different disturbances. The next section is devoted to application of multi-agent technology to this problem.

3 Application of MAS

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 programming techniques, it will generate the required plan. However working in a presence of disturbances 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 that lends to manufacturing system more flexibility to adapt 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 or several dynamics properties of MAS (e.g in [2]). We would like to add the following argument into this discussion.

The multi-agent system can be considered from a viewpoint of theory of finite-state automata. 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 completely deterministic way. If a control cycle is closed (see e.g. [2]) 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 own rules, there is no a central program that determines a states transition of the whole system. In the ”worst case” coupling n automatons with m states, the coupled system can demonstrate n^m states.

Evidently this ”worst case” has never to arise in the system, but how to control a behaviour of the distributed system without a central program (without a centralized mediator) ? The point is that all automata are continuously communicating in order to synchronize 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 currently solving problem. This is a main difference to the ”centralized programming” approach. The central program can react only 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, the ”centralized programming” approach restricts the multi-agent system although there are essentially more abilities to react.

The sufficient number of degrees of freedom represents a key problem of multi-agent technology. On the one hand if the system is hard restricted in the behaviour, the advantage of MAS is lost. On the other hand, if the system has too many degrees of freedom it can communicate an infinitely long time. In other words only several combinations of agents states have a sense and the point is 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 suggested here solution is based on a hierarchic software architecture that supports agent’s autonomy.

Before starting to describe an approach, it needs to mention one methodological point concerning decentralization of multi-agent system, shown in Fig. 3. The MAS solves a problem by using some methodological basis. For example the CS and CO 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, messages transfer, synchronization, decision making and so on. 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 ordered way, i.e. to include cooperation mechanisms (protocols) into this distributed description. In order to enable a transition from the interaction pattern to the *cooperation protocol* (see Fig. 3) a notion of a role is introduced [3]. A role is associated with a specific activity needed to be performed (according to a methodological basis). Agent can ”play” one role or a sequence of roles. In this way interactions are primarily determined between roles, an agent (with corresponding abilities) handles according to the role playing 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 agents organization, so that any modifications of agents do not affect the cooperation process and vice versa [3].

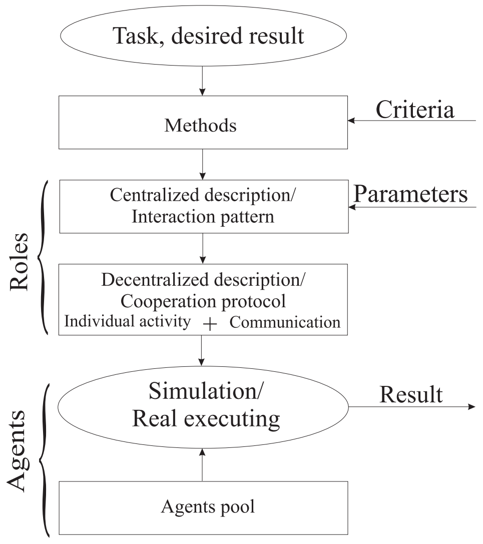
Task, desired result
Methods
Criteria
Centralized description/ Interaction pattern
Parameters
Roles
Decentralized description/ Cooperation protocol
Individual activity + Communication
Simulation/ Real executing
Result
Agents
Agents pool

Figure 3. *Methodological approach.*

The interaction pattern determines a *primary activity* (primary algorithm) of multi-agent system. The primary algorithm includes also some parameters whose modifications can be commonly associated with disturbances. Variation of these parameters does not disturb an activity of agents. In this case these are the expected disturbances, a reaction of the system on them is incorporated into the primary algorithm. However due to specific disturbances every agent can reach such a state that is not described by a primary algorithm and where a performing of the next step is not possible. In this case the agent is in emergency state and tries to resolve the arisen situation all alone or with an assistance of neighbour 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 emergency state is to change a part of the primary algorithm so that to adapt it to disturbances. The disturbances causing local emergency are expected (predicted) but not introduced into the primary algorithm.

The primary algorithm as well as its parametrisation is optimal only for specific conditions. If disturbances change these conditions the primary algorithm became non optimal and it has no sense to repair it. All agents have collectively to 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. Now we start to 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 sec. 2. Each working step in the approach is represented by a node in the constraint network shown in Fig. 2. 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.

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. 4 has three phases p_0, p_1, p_2 whose main result consists in determining a structure, neighbourhood relations and parameterization of nodes of the constraint network. The roles γ_0, γ_1 are "Initializers" of WS-

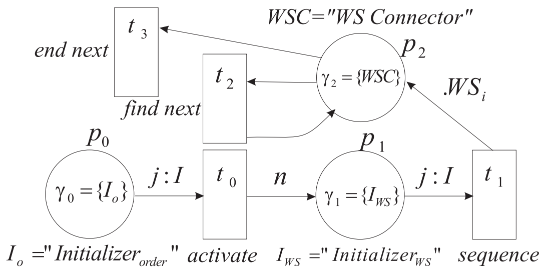

Figure 4. *Primary algorithm: Parameterization part.*

order and WS-nodes correspondingly. The role γ_0 is activated by the first production order, this role reads resource-objects and determines how much 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 parameter n as a 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 there exists no next node needed to be connected.

The propagation part, shown in Fig. 5, consists of

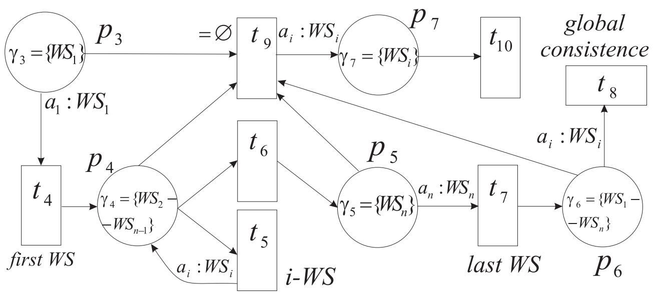

Figure 5. *Primary algorithm: Propagation part.*

three blocks: local (the phases p_3, p_4, p_5), 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 last nodes whereas γ_4 does the same for all other nodes. The transition t_7 proves whether the local propagation was successful for all nodes and activates then the global propagation in γ_6. We emphasize the local propagation requires a sequential executing of roles whereas in global propagation all roles can be in parallel executed. 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 restart of the local propagation.

3.2 The local and global emergency

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 executing 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 on. 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 point is whether a manufacturing is able to react by all alone, or it needs additionally a specific resources/abilities like a fire brigade.

The agent playing a current role cannot perform recognition of emergency. For this aim a so-called *activity guard* agent is needed whose macroscopic interaction pattern is shown in Fig. 6. In phase p_0 it observes an execution

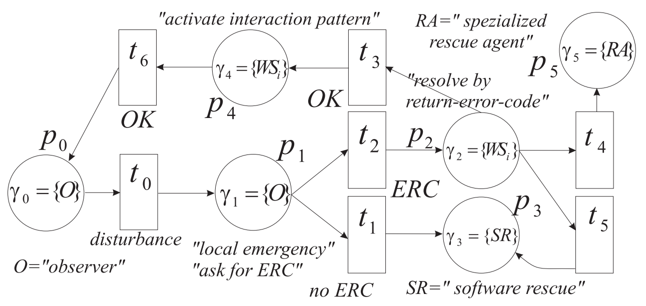

Figure 6. *Macroscopic interaction plan for local emergency.*

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 identifying the arisen problem. 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 resources ERC, activity ERC and so on. Based on this returned code a specific problem solving process can be started. If a 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 interaction patterns of a complex nature that are hierarchically called when the corresponding role is activated. 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 at the "time out" messages an agent cannot be able to form the error-return-code. This situation points to the case when an agent cannot accomplish internal activity cycle not because of absence of resources (i.e. problem-oriented emergency), but because of internal confusions 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 points to internal software errors. The second aim is to estimate the problems in external activity that lead to the "time out" emergency.

The idea of SR agent is based on the autonomy cycle shown in Fig. 7. The autonomy cycle represents the common steps that any agent cyclically executes. If there is an emergency of this kind it means that 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 by real one until the problem is brought to light. However there exists one critical point concerning the reaction of SR-agent on the detected problem. At least in the given implementation the SR-agent reacts only by sending a corresponding error-message.

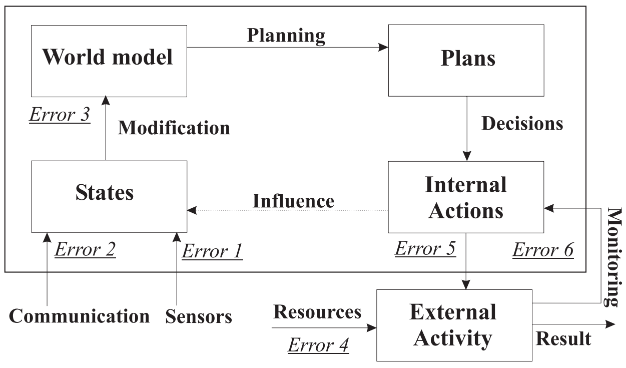

Figure 7. *Autonomy cycle of an agent.*

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 left effects. Firstly, they disturb the global criteria underlying methodological approach so that it is not more valid or, secondly, 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 so changed that the most restrictions will disappear. This leads to the agent's state space that after the CS approach is equal (or almost equal) to initial space, i.e. the original methodological assumption is not more valid. The mentioned disturbances has 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 [4].

The second reason causing global emergency is an irresolvable local emergency. Before to declare a state to be generally not resolvable, an agent (even rescue agent) transfers information about the problem to other agents in hoping they have the needed resources/abilities and can solve a problem. This solution may have also a local or global form. In 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.

Now, the point is of what kind of global change is required by global emergency in the agent group ? If the methodological approach is no more 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 not less important - how should it arise) ? Generally, there are two possibilities, either to use in advance made interaction patterns or to generate this pattern dynamically by a request. The generation of an interaction pattern represents a separated point that needs a description of required semantics, parsers and so on. Because of this reason as well as high complexity of such a generator it is a future outcome. If the measures undertaken in case of 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 approach described in the previous sections allow generating the sequence of working steps that satisfies all local constraints. However there are two important steps, that the optimization needs to be performed on. The first of them is an order of the working steps in the group 2 and 3 (see Fig. 2). This optimization can be performed by exhaustive search. The second point concerns the local decisions (concerning machine and position) made by agents. But the search space (taking into account the forecasting effect) grows in this case exponentially and e.g. even for 16 agent (2 production workpieces, forecast for next 4 positions) comes to $\sim 10^9$. Therefore exhaustive search methods like constraints optimization are inefficient even on very fast computer. The search space can be essentially reduced if to take into account the following observation.

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. 8. 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 case the optimal (or near optimal) scheme for the first iteration is found, next iteration can use the same scheme. The distributed approach being able to treat this kind of pattern-like problem is known as ant colony optimization algorithm (ACO) [5].

This method originated from observation of ants in the 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. Similar strategy can be applied to local decisions of agents, participating in the plan making.

Agents after the CS approach choose several assignment plans from the generated set of them and form an optimization pool. These assignment plans can represent also only segments of plans (these connected working steps represent independent parts of 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 are included into this pool, the 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 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 has always solutions with a high pheromone rate, from them the most optimal one will be then chosen. Moreover

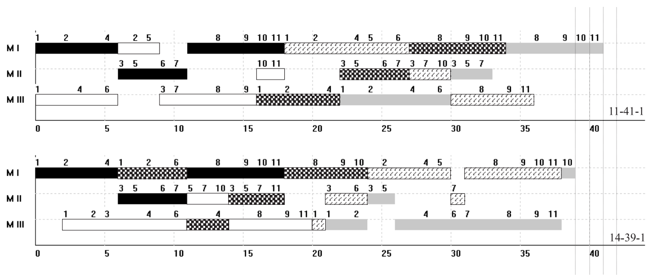

Figure 8. *Assignment plans with different length and costs.*

combining the near-optimal plans in the top of the pool, the assignment plans with integral optimization criteria (compromise schema between length and cost) can be achieved, as shown in Fig. 8.

5 Conclusion

The presented approach enables to react reasonably to disturbances in manufacturing by using an agent-based approach. It does not require any centralized elements, that essentially increases a reliability of common system. Several problems such as a program generator or distributed learning still remain unsolved and are destined for further investigations.

References

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

[2] G. Weiss, Multiagent systems (MIT Press,1999).

[3] M. Muscholl, Interaction und Kooperation in MAS (Stuttgart: PhD thesis, University of Stuttgart, 2001).

[4] O. Kornienko, S. Kornienko, P. Levi, Collective decision making using natural self-organization, Proc. of CIMCA'2001, Las Vegas, USA, 460-471.

[5] D. Corne, M. Dorigo, F. Glover (eds.), New ideas in optimization (McGraw-Hill, 1999).