Reactive Agent Mechanisms for Manufacturing Process Control

Kasper Hallenborg and Ask Just Jensen and Yves Demazeau

University of Southern Denmark, Maersk McKinney Moller Institute, 5230 Odense M, Denmark,
Email: {hallenborg;just}@mmmi.sdu.dk
CNRS, LIG Laboratory, 38000 Grenoble, France, Email: Yves.Demazeau@imag.fr

Abstract: This paper proposes a multi-agent approach to control chemical processes in a production system, where items should undergo chemical reactions in different baths. Recipes for the processes only specify minimum and maximum times for each bath, and the recipes depends strongly on characteristics of the item.
The PACO paradigm for reactive agents has been applied to create the control software for the system, which is based on simple physical force-like interaction mechanisms, so the agents manage to fulfill their individual and global goals. An outline the problem is followed by an introduction to the PACO paradigm and arguments that explains how PACO suits this problem. Detailed explanations of how the agents are designed and their interactions are given as the core contributions, followed by results of the approach in real world scenarios.
The investigated problem is part of a larger research project, DECIDE, focusing on applying multi-agent control in production systems. Part of the work have previously been presented in [15].
Keywords: multi-agent system, PACO, interaction mechanisms, flexible manufacturing

1. Introduction

Most industrial companies have experienced radical changes in market demands in recent years. Mass series and standardized products, have been replaced by order based production and a high degree of user customization, which falls back on the production system as strong requirements for high flexibility. In some setups flexibility could be meet by low switching times, but ideally the system should be able to handle a large variety of item concurrently in an efficient way. That requires new control algorithms, as dedicated hardware would be optimized for a specific production. Global optimization is typically NP-complete or inappropriate due to a dynamic production environment, where constant changes will lead to continuous replanning. Thus flexible manufacturing systems have been a focused research area for decades [4, 5, 6, 13, 18], and multi-agent technologies have been a natural approach for the control software of such systems, as multi-agent systems have a ingrown focus on flexibility. Whereas most research have focused on rather flexible setups of the production environment and aimed for shop floor production and low order series. Traditional flow oriented production facilities are also candidates for increased flexibility, if the control logic is exchanged with a more dynamic solution, where agents could play a central role. Thus agent based systems can close the gab between dedicated hardware systems and handmade items, and illustrated in Figure 1.

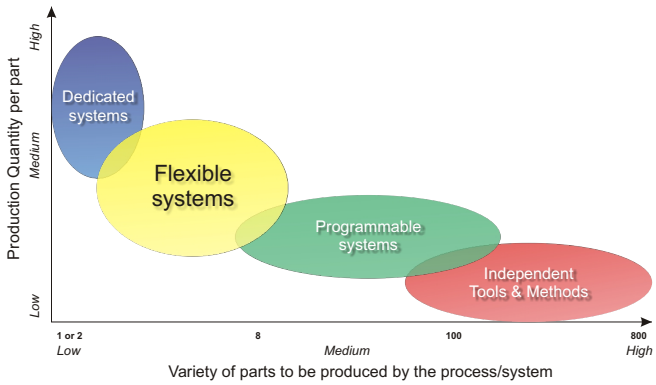

Fig. 1. The manufacturing flexibility spectrum, adopted from [3]

This paper will present such a production setup, which in principle are bound to a rather strict and determined production process, but demands for a high variety of orders adds the challenges of flexibility that might be solved by a local agent-based optimization.

The paper is organized as follows. First the general problem of the researched case is discussed. Next it is related to previous work, and the PACO paradigm is shortly described, which is used for the agents in this case. Details of how PACO agents are organized, modeled, and implemented to solve the problem at hand follow, before results of real world scenarios and added improvements are presented. Finally some concluding remarks will bring ideas for future work of this ongoing research work.

1.1 The problem

Timetabling of classes is a classical constraint satisfaction problem, which is known to be hard or even NP-complete for large schools or universities. But imagine the increase in complexity if duration class sessions where allowed to vary dynamically in runtime. The argument could be that to take full advantage of the resources (teachers and classrooms), teachers should only stay as long as required for the students to understand the topic, but bounded by a minimum and maximum timeframe.
Similar dynamic train or bus tables would be a real challenge, but might be a way to increase capacity of the overall transportation system with fewest resources.

The researched case is conducted in collaboration with Denmark's most well-known manufacturer of high-end audio and video products. The products are respected worldwide for their extremely high-quality finish and design, and the investigated production facility is the process, which gives the surface of the product the high-quality finish. The process is known as an anodization process that increase the corrosion resistance of aluminum, but coloring of the surface is also part of the chemical processes. In a generalized and simplified form, the problem could be described as a number of chemical baths, which the items have to visit according to a prescribed recipe. Besides containing information about which baths to visit and in what order, the recipe also give an allowed timeframe for the item to stay in each bath. Items are grouped on bars with the same recipe, but a mix of different bars (= different recipes) could be processed at the same time in the production system.

The system consist of about 50 baths, and a typical recipe would have roughly 15-25 baths to visit, even though all recipes do not have to visit all kind of baths, there is still room for additional baths of the same type to overcome bottlenecks as the processing times in the baths types varies a lot. Thus the recipe contains only bath-types not bath-number and it the task of the control software to allocate a specific bath among duplets for every bar.

Fig. 2. A crane moving a bar in the system

Three slightly overlapping cranes move the bars from one location to another in the array of baths, as illustrated in Figure 2. Here a simplified notion for the movements will be used, but in practice they are more complex than that, because moving between some specific baths include sub-processes, such as rinsing the bar of items, open and/or close the lid of a bath, etc., but it comes down to an estimated travel-time of moving a bar from bath u_j to bath u_{j+k}. In general the cranes are not considered to be a bottleneck in the production system, as they handle the tasks quite sufficient. Apart from the baths and cranes, an important part of the system is the input buffer, where typically around 30 bars are waiting to be processed, which also is an important focus point for the control software, because choosing the best bar to fit the current configuration is the key to optimize through-put. A general overview of the system is presented in Figure 3.

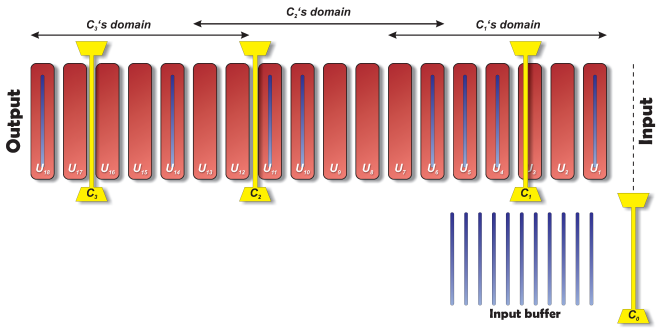

Fig. 3. Generalized overview of the system for the anodization process. Bars move from right to left

At first sight the problem described seems to be a candidate for classic optimization and scheduling principles, but as mentioned above the allowed timeframe for processing[1] each step of the recipe complicates the task. Another issue is the dynamical production environment, which has great impact of the system's ability to recover and finish the current bars, but also run as best as possible under partly breakdowns. Examples of unpredictable error conditions could be that the temperature of a bath is too low and must be heated before being available again, crane breakdowns, the liquid level of a bath is too low, rapid orders, etc. An agent based approach must have focus on the dynamics and be able to recover or continue as best as possible under such conditions.

1.2 Related work

The problem is classic - the throughput of a system should be optimized, by optimizing the flow between sub-processes and handle the inflow process correctly to utilize the system as much as possible.

In abstract terms there exists a number of tasks, q_i for $i = 1, 2, \ldots, n$, with k_i subtasks[2], $q_{i,1}, q_{i,2}, \ldots, q_{i,k_i}$. The subtasks are interconnected and the order cannot be changed, and should be handled as visits to processing stations - determined by the recipe. The agents, which could be the baths and cranes, must coordinate their local activities for the global plan to be optimal. From a modeling point of view the TAEMS[3] Framework [9, 16] would be an obvious choice as a modeling tool to describe the structure of the plan. The TAEMS modeling approach shows it's real strength for taskgroups with less coupled or interdependent tasks.

GPGP is a set of generic coordination mechanisms for cooperative agents in a task environment proposed by Keith Decker [10]. GPGP is a generalized extension of the PGP (Partial Global Planning) algorithm, which is capable of handling deadline for the task of the agents. GPGP could be applied in this case as well, but the homogeneity of the task structures and task flow-oriented approach for the closed system of the case, appeal for other approaches as well.

Clement and Barrett also introduce the Shared Activity Coordination (SHAC) [7], which provides a general approach for

[1] Only minimum and maximum times are given for each step of the recipe
[2] note that the number of subtasks might be different for each taskgroup
[3] Task Analysis, Environment Modeling and Simulation

interleaving planning and exchange of plan information among agents based on shared activities.

Other approaches exists as well, but in general they are focused on the agent performing different tasks in the environment, where the tasks can be regarded as atomic actions. That would make the baths and the cranes to agents, but in this case the tasks are not atomic in the sense that duration of a task in not fixed.

In principle the planning approaches search a solution space to find an appropriate valid plan to a constraint satisfaction problem (CSP). There is no guarantee that the plan is optimal, as the problem often are NP-complete. In this case the task durations varies between the minimum and maximum times specified in the recipes, which gives us inequality constraints that complicates the problem even more. In traditional planning we try to find an optimal plan among valid plans, but using the PACO approach we strive for an valid plan using simple reactive mechanisms on a configuration that might be optimal, but not valid.

Heuristics and approaches for some combinatorial optimization problems, such as the use of internal forces in elastic nets algorithms for the classic travelling salesman problem [12], is also somewhat similar to the forces applied in PACO that will be described next.

1.3 PACO approach

PACO is a contraction of *coordinated patterns* [11] and takes a simple approach of designing the agents. PACO focuses on reactive agents situated in an environment, where all agents are considered as partial solutions of a global problem [14].
Interactions between the agents and the environment are generally applied and modeled as forces, and by giving the agents a mass, they will at least from a conceptual point of view have both velocity and acceleration, which is valuable when adjusting the priorities between interactions. The applied forces are spring-like forces, which reduce the risk of oscillating interactions, but also secure that the system will converge to an equilibrium state at some time in the coordination process between all agents.

The PACO paradigm states that agents are purely reactive, thus they do not hold an updated internal representation of themselves, other agents, or the environment, so they have to respond to all change of the environment in which they are situated. This general idea suits the researched case very well, as agents after an initialization process will hold some kind of plan for handling the current set of bars in the system. Whenever a new bar is introduced, or some kind of unforeseen or expected events happen within the system, such as a crane breakdown or a bath needs cleaning, it is just a new stimuli to the agents of the control system, and they will start searching for a new equilibrium state through their interactions.

Each agent under the PACO paradigm is defined by three fields, which divides the agent model in three coherent components

Perception field determine what the agent can perceive about its environment.

Communication field determine which agents an agent can interact with.

Action field determine the space in which an agent can perform its actions.

From a system point of view the PACO paradigm also splits the system into conceptual parts, which follows the VOWELS formalism [8]. Basically the VOWELS formalism decomposes the problem into four components of the MAS domain of a system, described by the vowels-initiated concepts

Agents are the classic entities to consider, when developing a MAS system, as agents are determined to be the local actors carrying out the tasks of the system. Agents can be described by the classic characteristics introduced by Woolridge [19]; *autonomy*, *social ability*, *reactiveness*, and *proactiveness*. Especially *autonomy* is in focus here to emphasize that agents have goals and are self-determined.

Environment is the space in which the agents exists, moves, and interacts. The space could be both informational and conceptual, but typical the environment is represented by a model of the physical space of the MAS community.

Interactions and communication are evident in MAS, due to the aspects of distribution of such systems, and originally introduced by Woolridge as the social ability of a agent [19]. Interactions in the MAS community could take many forms; negotiation, collaboration, coordination, queries, or generally any kind of information exchange between agents, which have tried to be supported and standardize by query languages, such as KQML, or interaction specifications, like FIPA's interaction schemes. Interactions could be formed as abstract as speech interactions, but the PACO paradigm usually apply interactions as forces, spring-like or electrostatic forces, as mentioned above.

Organization: Similar to humans, agents can benefit from being organized, either explicitly defined in classic organizational structures, or the organization could emerge from simple interactions among the agents. Organization often serves the purpose of grouping agents with similar or related actions or behaviors as usually exploited by PACO. Organizations can be helpful to support agents in planning, performing actions, requesting information, or realize global goals of the agent system [17].

2. Agent design

This section describes and discusses how the PACO approach under the VOWELS formalism has been applied to the researched anodization system. The following subsections cover each part of the method

2.1 Environment

Before agents can be created and assigned with goals and behaviors there need to be an environment for them to exist in. In this case the environment is the baths and cranes. The environment is modeled as passive resources, which the agents can ask about their status and book for a given time. Baths are accessed through a bath controller, which makes baths of the same type look as only one bath, capable of containing more than one bar at a time. Theses baths can be asked about free space in a given direction by an agent or about whether or not a free time slot of a required timeframe exists in a

specific period. If the space is occupied, then the bath can tell, which agent is blocking. A bath has no possibility to prioritize or assign time to individual agents as well as pushing or cancelling time already assigned. It is the responsibility of the agents to fight for time slots themselves.

2.2 Agents

The recipe for each bar of items is split into a number of agents. One agent is created for each step of the recipe, and all agents made from the recipe forms a group. An agent is born with some knowledge, as it knows which kind of bath it must visit, it holds the allowed minimum and maximum time to stay in the bath, and it knows its predecessor and successor agent of the group. It does not know the rest of the agents in the group and it has no way of communicating with them. Thus the scope of the agent within its group is rather limited, which simplifies the interaction model.

To succeed, an agent must visit a bath of the right type, but not necessarily at the right time. The agent has a size equal to the time slot it occupies in the bath. Therefore by its representation, agents can be seen as physical manifestations of the problem in focus. Two bars q_i and q_j, split up into two groups of agents, $[q_{i,1}, q_{i,2}, \ldots q_{i,n}]$ and $[q_{j,1}, q_{j,2}, \ldots q_{i,m}]$ added to the virtual model in random places it could look like Figure 4.

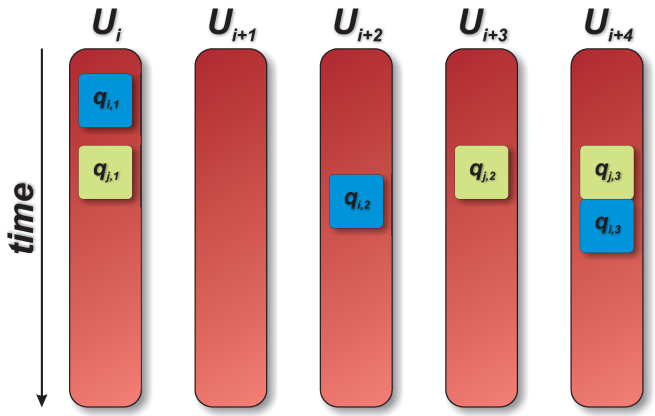

Fig. 4. Three agents from each agent group occupying timeslots in the baths

2.2.1 Perception, communication, and action fields

As stated earlier the PACO paradigm defines three delimited fields of how PACO agents experience the world; the perception, communication, and action fields.

The perception field consist of the predecessor of the bar, as the movement of that agent affects the forces (described below) applied to an agent. Furthermore the agents above and below (in the time domain) that want to visit the same bath are also observed, to avoid overlap of agents in the same bath.

The communication field solely consist of the predecessor, as it should be notified if the agent could meet its goals.

The action field consist of the baths of the requested type, and organization secures that an agent only see one particular bath, even if the bath type is duplicated in the system.

2.2.2 Agent goals

An agent has two main goals:

- Go in the right bath
- Stay close to the predecessor agent of its group

When both goals are satisfied, for all agents of a group, the bar represented by the group has a valid way to be processed by the system. Furthermore an agent has some constraints:

- Keep distance to both the min and max time
- Help the successor to stay close
- Help other agents in same bath type to fulfil goals

Constraints is added to make agents cooperate with others in fulfilling there goals too. When agents from two groups share interests to the same time slot for a given bath they have to be able to negotiate about, who will win the timeslot.
Therefore an extra type of agent is introduced, an observer. For each group of agents one observer agent monitors the movements and how satisfied agents are in general. Information withdrawn from this observer is used when solving conflicts.

2.3 Interactions

Agents move around in the virtual world in discrete steps. They calculate a force vector $\boldsymbol{v}$ as responses to input/output from the three fields.

Each discrete time step has two parts, first all agents gets a parallel chance to decide which way to go and at what speed. Hereafter they get the chance to move themselves. In this moving step they will try to move in the direction and distance specified by $\boldsymbol{v}$, within the space allowed by their action field.

2.3.1 Basic forces

The most basic behaviors of the agents come from their primary goals and are modelled with two forces; a spring force and a gravity force.
The spring force represent the attraction to the predecessor, if any, and attracts the agent towards the point where the predecessor's timeslot of the previous bath ends, so the bar can move from one bath to another, which is a criteria for a plan to be valid.
A spring force is denoted: $\boldsymbol{F} = -k\boldsymbol{x}$. $\boldsymbol{F}$ being the force, k the spring constant and $\boldsymbol{x}$ the distance. This gives: $\boldsymbol{F}_s = -k_{parent}(\boldsymbol{x} - \boldsymbol{x}_p)$.
Where k_{parent} being a static constant, $\boldsymbol{x}$ the agents position and $\boldsymbol{x}_p$ the ending point of the predecessor.

To make the system stay in motion, until a valid equilibrium is reached, an extra parameter, a predecessor multiplier (p_m), is added. Each time a spring force is calculated the agent will check if it has come closer to its predecessor. If it has not, it will increase p_m, to enhance the attraction force

$$\boldsymbol{F}_s = -k_{parent}(\boldsymbol{x} - \boldsymbol{x}_p) \cdot p_m \quad (1)$$

p_m is bounded. It can newer go beneath 1 or above some fixed value, in this case 3 have proved to be enough. It is increased with the same percentage each time the agent has not come closer to its goal and is decreased again with the same percentage, when the agent starts moving.

The second force, the gravitational force, tries to pull the agent up. Up in the virtual model represents beginning of time

in the real world, as shown in Figure 5. The gravitational force is given by: $\boldsymbol{F}_g = m\boldsymbol{g}$.
Where m being the mass of the agent and $\boldsymbol{g}$ the gravitational acceleration. With the mass of all agents being the same, it gives $\boldsymbol{F}_g = \boldsymbol{k}_g$, where $\boldsymbol{k}_g$ is a static constant force vector. This gravitational force is only applied to an agent when it is floating freely. If the agent is in contact with another, in the direction pulled by the force, the counterforce from the contact will cancel out the gravity.
The total force is denoted $\boldsymbol{F}_t$

$$\boldsymbol{F}_t = \boldsymbol{F}_s + \boldsymbol{F}_g \tag{2}$$

With only these two simple forces, a set of agents can be added and align them self and thereby make a valid schedule of how to be processed by the system. See Figure 5.

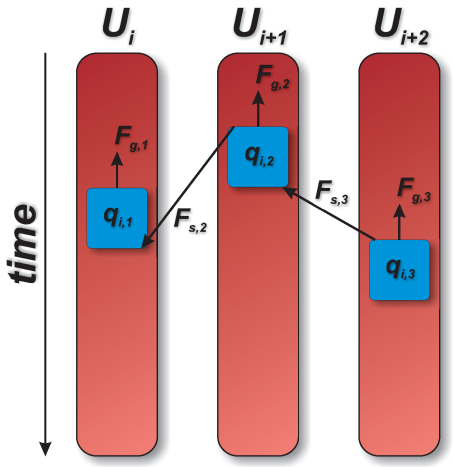

Fig. 5. Illustration of the basic forces for the agents

The spring force serves to compact the plan of an agent group, in order to minimize the total processing time of a bar, whereas the gravity force works to compact the entire plan for all bars in order to maximize utilization of all baths.

2.4 Organizations

To make the interaction between the agents more flexible a number of social laws are introduced:

Law 1: If there is a certain amount of free space around the agent, increase size to

$$T_{current} = T_{min} + X(T_{max} - T_{min}) \tag{3}$$

X being a static constant and $T_{current}$, T_{min}, and T_{max} being respectively the current, minimum and maximum time slots of the agent.

Law 2: If another agent, using the same bath, approaches within a given distance, then shrink the current size until T_{min} plus a given margin is reached.

Law 3: If the successor is unable to reach its second goal, then stepwise increase $T_{current}$ until T_{max} is reached.

2.4.1 Options for problem solving

If an agent needs to go in a direction blocked by other agents, it should be able jump over, push or switch place with one of the blocking agents, as illustrated in Figure 6.

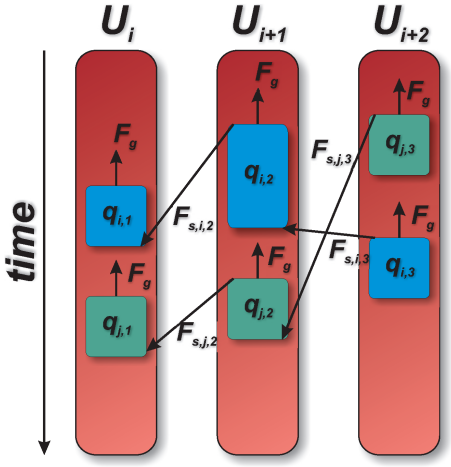

Fig. 6. Illustration of a conflict between two agent groups

For this purpose three laws are introduced. They are respected when agent A wants to go in a direction blocked by agent B.

Law 4: $\boldsymbol{F}_t$ for A is greater than the current size of B, and a time slot of at least A_{min} is available between the end of B and the length of $\boldsymbol{F}_t$, A jumps to the other side of B, without notice.

Law 5: If there are no room for A on the other side of B, but B is trying to move in the opposite direction, and if the size of $\boldsymbol{F}_t$ for A is greater than half of B_{min}, then they switch place.

Law 6: If none of two previous laws have applied, but A still wants some or all of the time slot assigned to B. A starts a negotiation based on the general satisfaction of group A and B[4]. If A wins this negotiation, B is pushed away, otherwise they will have to stay.

With this set of forces and laws for solving problems a decent number of bars should be able to be split up into PACO agents and added to the system. Hereafter they will align themselves in a roughly optimal way or stay in motion trying to seek their goals.

3. Discussions

Apparently, the real challenges of applying the PACO paradigm to an agent-based control system like the problem presented above is designing and fitting the forces used for interactions. This section will discuss some of the experienced issues for the case, and describe how they have been solved through adjustments and the extra laws introduced above.

The focus will primarily be on two matters, which had the greatest impact on problems experienced in the solution based only on the basic forces.

[4] Information about the satisfaction is withdrawn from the attached observer agent

3.1 Avoid competing agents within a group

The agent group, which spawns from the creation of agents for a single bar of items, is not modeled or implemented as a sole entity in the system. Thus no overall goal or intensions of the group can be directly implemented, but must be realized through the aggregation of sub-goals met by the agents within the group. The tension appearing inside a group due to the spring forces of the agents, can to some degree lead to competitions among agents within a group, but the flexibility laws presented in section 2.4 make it easier for the system to reach an equilibrium and dampen the inter-agent tensions. Particular laws 1 and 3 are added to cope with these side effects of the basic forces. Law 1 simplifies the process of attraction and stabilize the movements of a successor agent to its predecessor, due to the expansion of the current timeslot for an agent in a bath, if it is too hard to pack the schedule for a bar tighter. Note that the plan for each bar at the end must form a consecutive sequence of visits to baths, as the cranes move bars from bath to bath, because the system has no spare slots that temporarily can hold a bar. Whereas law 3 more directly connects the plan of a group and increase robustness in the coordination process.

On the other hand law 2 is important as well even though it is orthogonal to the agent groups, but it adds flexibility by minimizing the slot time requested by an agent. It is not a direct coordination mechanism between agents from different groups, but allows some mutual impact on their actions.

3.2 Handling oscillating task shifts and stick to suitable schedule slots

Without doubt the most challenging part of optimizing the overall plan for the system is to decide when and how conflicts between agents should be solved. No method or measurements exist to validate if a current configuration is optimal or jumps between the agents should be handled. Clearly from law 4 and 5 of section 2.4.1, trivial conflicts are handled without contracting classic local optimization principles. Especially to avoid oscillating shifts between agents from different groups with interest in the same bath, law 6 serves the purpose of dampen inter-group tensions.

Relying on the aggregated information, collected by the group observer, would increase the abstraction level and understanding why and how agents should shift in case of a conflict, in order to maximize utilization and meet the global goals. Oscillations in general occur when the current configuration is jumping between local minima for an undamped system. So a jump in law 6 would only be allowed if the satisfaction parameters measured by the observer are expected to increase.

4. Results

In order to validate the PACO paradigm for the agents, the control system has to be tested to see if it can create valid plans for the bars. Which is done by measuring a satisfactory rate for an agent group. A full valid plan would have 100% satisfactory rate, which means that for a given bar all visits to baths in the recipes comply with the minimum and maximum timeframes, and that moves between two consecutive visits are connected with no glue time.

Naturally an indicator of the strength in the approach is the dynamic reactiveness, because the core problem is a classical optimization problem, but as it is NP-complete, the system do not have the time to find an optimal solution in a real world scenario. Thus the number of time steps required for the system to find an acceptable solution is of primary importance.

In Figure 7 the average computation time required per agent to calculate the forces and move the agents are shown. Naturally the computation time increase significant for scenarios with more than one bar, but in all cases it stays below 0.5 ms per agent on an average PC. It is reasonable efficient, as we can see from Figure 8 that the agents relative fast move to a rather stable level of their satisfaction rate.

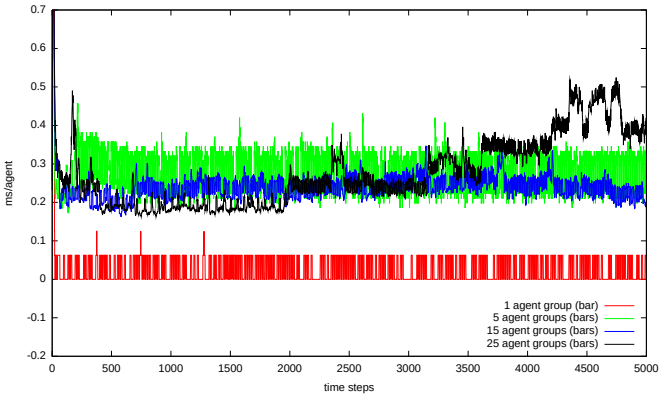

Fig. 7. Average computation time per agent per time sted for scenarios with 1, 5, 15, and 25 agent groups

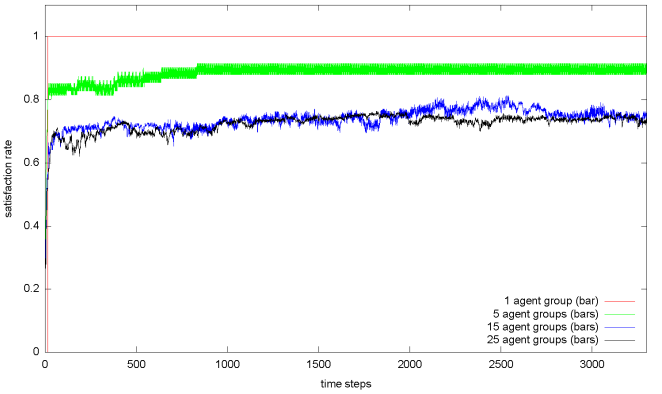

Fig. 8. Satisfaction rate for test with 1, 5, 15, and 25 agent groups

Figure 9, which is a zoom of Figure 8, also shows that valid plans (satisfaction rate = 100%) are only created in the 1 group scenario. From the graph of the 5 group scenario it is easy to see that some fluctuations is present around a level of 90% satisfaction. For 15 and 25 groups the picture is a little more blurred, and more mechanisms should be applied to meet the 100% level.

Finally in Figure 10 is shown that the end-time (when it have visited all requested baths) of the last bar is decreasing, so the algorithm is creating more efficient plans. Jumps between agents during the planning process will naturally leads

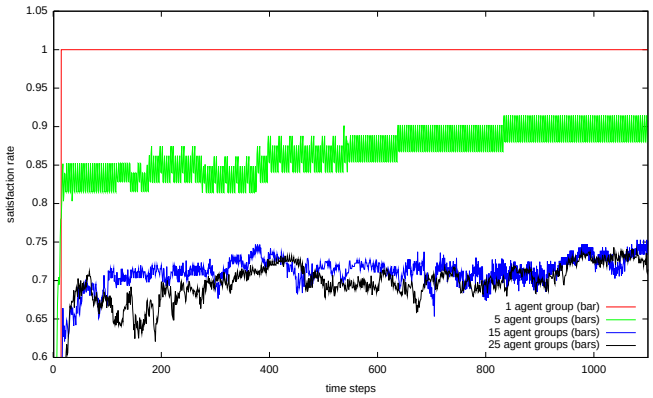

Fig. 9. Satisfaction rate for test with 1, 5, 15, and 25 agent groups

to increased production time before the plans starts to settle again.

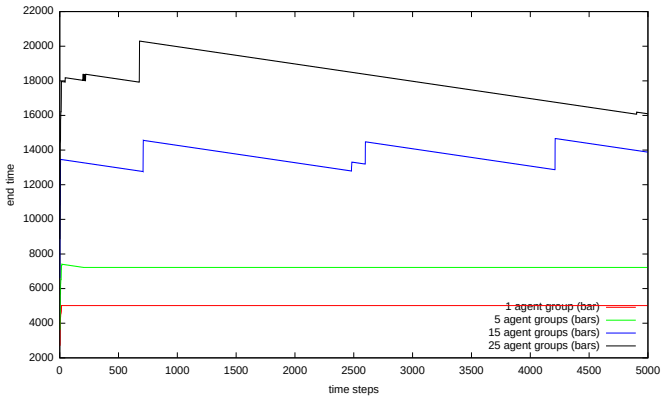

Fig. 10. End time for the last bar of scenarios with 1, 5, 15, and 25 agent groups

5. Fine-tuning improvements

Valid plans are not meet in all scenarios, as the results of the previous section show. In order to improve the results a number of experiments have been conducted, which clearly improve the number of valid plans being generated.

Each of these strategies will be described in the following sections

5.1 Active, sleeping, and locked agents

During tests it has generally been observed that agents end in a deadlock situation, where one or more agents oscillates, as illustrated in the constructed graph of Figure 11

In the example bar 2 and 3 are stable at full satisfaction, and bar 1 is oscillating, but there is no guarantee that the plan for 2 and 3 are optimal, so to bring the agents out of the deadlock, it could be chosen to push bar 2 and 3 into play again, or accept and lock their positions.

Thus, a promising approach is to give the agents a state, which determine their ability to perceive the environment, and how they should react to new stimuli. Three state are obvious for the agents

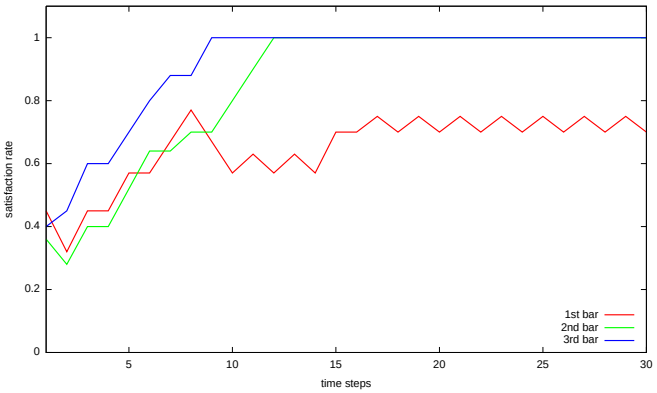

Fig. 11. Constructed example of agents in a deadlock situation

Active agents are agents that observe everything of their fields, and fully accept the input and influence of other agents according to the laws described above. In other words active agents behave as expected from the design section.

Sleeping agents are agents, which no longer possess intentions to move, and which other agents no longer can expect to influence. Anyway stimuli from the environment and input from other agents can grow so strong that the agent is awaked again.

Locked agents are agents that no longer are under influence of the environment or other agents, but internally they can still decide to unlock and become active again.

The interesting issues are the transitions from state to state. For a sleeping agent there is two options, either does the agent itself or its group expect that it can improve its position, or the agent is pushed to hard from the environment. Again in the later case, the pressure could be on both the agent itself or the group it belongs to.

An agent could expect to improve its position if free space above it has become available, and not necessarily directly above the agent. Also if a larger chunk of free space have become available earlier in the time domain.

The pressure from other agents can be controlled by a threshold value, so the agent are awaked if the forces applied from other agents sum too high. Equation 5.1 gives the summed force of impact from other agents,

$$\boldsymbol{F}_{I_O} = \sum_i \boldsymbol{F}_i S_i d_i \tag{4}$$

where $\boldsymbol{F}_i$ is the force from the i'th agent that want the position, S_i is the satisfaction rate of the i'th agent, and d_i its distance in time to the sleeping agent.

Also the pressure on an agent from its group can be expressed as a summing force that can break a threshold value. Equation 5.1 gives in-group tension force that can bring the agent awake again

$$\boldsymbol{F}_{I_G} = \sum_j \boldsymbol{F}_j \frac{1}{d_j} \tag{5}$$

Where $\boldsymbol{F}_j$ is the applied force from the j'th agent in the group and d_j its distance in steps to the sleeping agent.

Given those transitions an agent could fall sleep, if it is has found a steady-state and is not in conflict with other agents. Also if its group has found a stable level, but some members are oscillating.

5.2 Snapshots

Similar to some heuristics of classical constraint satisfaction problems an approach could be to back-track in the search space, if the overall solution seems to go in a wrong direction. So like a transactions mechanism a snapshot of a current plan can be taken and then allow the system to continue a defined number of steps in the current direction. If no improvements can be measured then the plan is rolled back to the snapshot and with modified parameters it continues in another direction.

Snapshots is a possibility, but it may contradict with overall design criteria of simple reactive agent. Especially if the agents have to consider the potential improvements of moving in a certain direction. Also the issue of when to take the snapshot is not trivial.

5.3 Predecessor validation

An agent adjust its position according to the free space around it, but also under influence of its predecessor's position. Experiments have shown that especially during the initial settling time of an agent group, some agents were strongly influenced by their predecessor agents, which were not very reliable with respect to their final position.

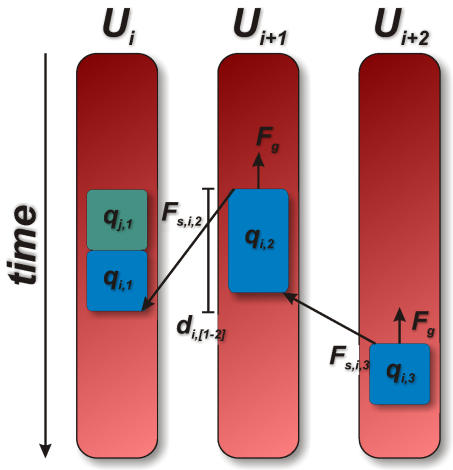

Fig. 12. Illustration of the parent validation problem

In the example agent $q_{i,3}$ would move upwards in time due to gravitation and the position of $q_{i,2}$, but it is rather obvious to see that $q_{i,2}$ is not very reliable due to position of $q_{i,1}$ that seems to have settled next to $q_{j,1}$.

It is not certain that the parent validation will improve the plan of the system, but it is introduced to dampen oscillations of agents. One way to validate the parent is to look at its position according to its parent, as illustrated by the dimension in Figure 12.

Figure 13 shows the result of a scenario with 25 bars both with and without the parent validation. As expected the plan becomes more stable with the parent validation, but it also have a longer settling time as a natural consequence.

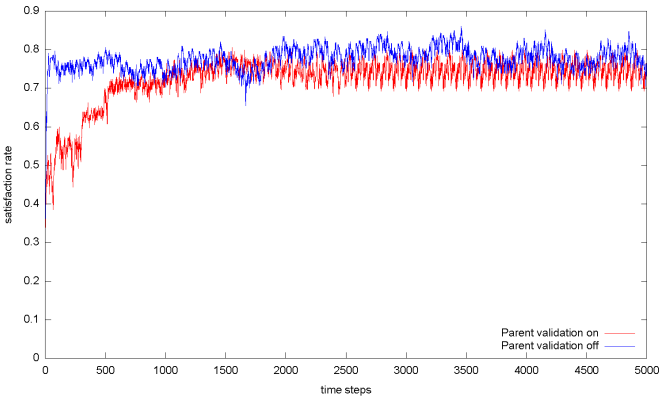

Fig. 13. Satisfaction rate for 25 agent groups with and without parent validation

5.4 Floating

The final improvement discussed in this paper is the concept of floating agents, which basically can be described from Figure 14

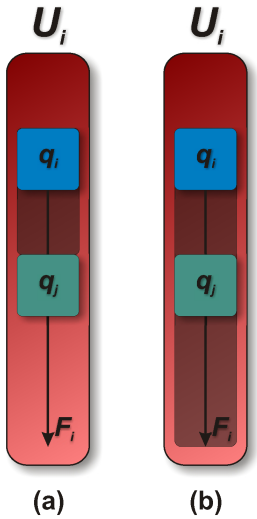

Fig. 14. Illustration of the floating improvement - adjusting the action field

According to the design described in the previous sections an agent q_i would behave as in the left case (a). The action field of an agent only allows an agent to move in direction of the resulting force until it is blocked by other agents, in this case q_j, even though the force is larger. It may lead to a conflict that can be solved by the laws described in section 2.4.1 in the next time step.

By extending the action field to the size of the resulting force and allow the agent to float over another agent g_j many conflicts might be avoided. That is similar to allow the agent to search for a valid position from the bottom of is action field, as illustrated in the right case (b), whereas the agent in (a) searches from the top until it meets a block or the end of the force vector.

Figure 15 present the results of the scenario with and without the floating improvement enabled. There might be more fluctuations with floating enabled, but as expected it dramati-

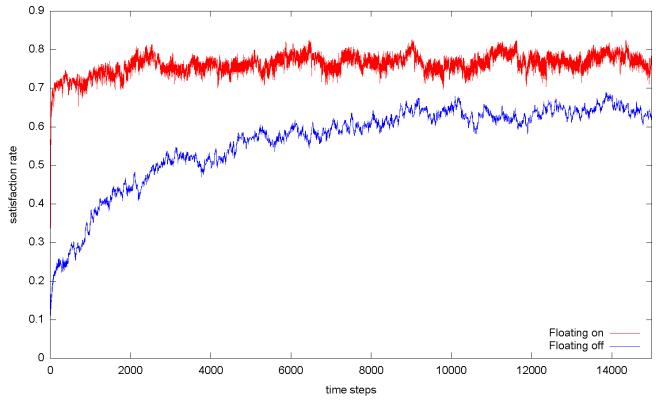

Fig. 15. Satisfaction rate for 25 agent groups with and without floating enabled

cally improves not only the settling time. Also the overall satisfaction level is increased significantly, as the agents avoid many conflict caused by tensions between agents.

6. Conclusions

This paper presents important research contributions from an application project under the DECIDE project, which focus on multi-agent control in production systems. The PACO paradigm have been applied, which suits the low-flexible setup of the given production environment. Besides designing the interaction mechanisms for the case, the model have been extended with a number of laws to cope with general problems and conflicts among the agents. In addition to that a number of strategies have been introduced to improve the performance of the control logic.

The approach is radical to many other paradigms of MAS in production environments, such as GPGP, as agents are not modeled as the resources of the environment, such as robots, baths, etc. Here agents originate from the real issues at hand. The issue of optimize utilization of bars by scheduling and fitting different bars in an appropriate way, which has nothing to do with either crane or bath capacities, as these factors are merely constraints that could be easily checked. The beauty and strength of this approach comes from the simple laws controlling the agents. All constraints from cranes and baths are extracted into a model of the environment. Thus all decisions and optimization arise from real simple, real-world inspired forces.

The presented model and design are generalized from the specific case, so the results would easily map to similar problems that can be modeled with reactive agents.

7. Future Work

Future work on the case is expected to contribute in a number of ways. Besides fine-tuning the improvements of the previous section, other approaches might be effective as well.

On particular issue would be to investigate the impact of the extending the scope of the agent interactions, so forces do not only exists between successors and predecessors. Also interactions between non-consecutive interests to a bath could influence agent intensions of sticking to a bath or forming multiple jumps.

In the future it would also be nice to compare the approach of PACO agents with more classical optimization methods, both regarding the results and the number of iterations required to find a workable solution.

Acknowledgements

Thanks to all the participants of the DECIDE project, which has been supported by the The Ministry of Science, Technology and Innovation in Denmark. A special thank to Bang & Olufsen [1] and Simcon [2] for their support, feedback, and creation of the simulation model of the system.

References

[1] Bang & Olufsen, `http://www.bang-olufsen.dk/`, 2007.
[2] Simcon, `http://www.simcon.dk/`, 2007.
[3] R W Brennan and D H Norrie, From FMS to HMS, Springer, 2003, pp. 31–49.
[4] S Bussmann, N R Jennings and M Wooldridge, Multiagent Systems for Manufacturing Control, Springer, 2004.
[5] S Bussmann and K Schild, An agent-based approach to the control of flexible production systems. 8th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2001) Antibes Juan-les-pins, France, 2001, pp. 169–174.
[6] F T Cheng, C F Chang and S L Wu, Development of holonic manufacturing execution systems, In Manufacturing the Future - Concepts, Technologies & Visions. Advanced Robotic Systems International, 2006, pp. 77–100.
[7] B J Clement and A C Barrett, Continual coordination through shared activities. AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems New York, NY, USA, 2003, ACM, pp. 57–64.
[8] J L T da Silva and Y Demazeau, Vowels co-ordination model. AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems New York, NY, USA, 2002, ACM Press, pp. 1129–1136.
[9] K Decker, TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms, Wiley InterScience, 1996, ch. 16, pp. 429–448.
[10] K Decker and J Li, Coordinated hospital patient scheduling. In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98) 1998, pp. 104–111.
[11] Y Demazeau, Coordination patterns in multi-agent worlds: Application to robotics and computer vision, In IEEE Colloqium on Intelligent Agents. IEEE, 1991.
[12] R Durbin and D Willshaw, An analogue approach to the travelling salesman problem using an elastic net method. Nature, Vol. 336, 1987, pp. 689–691.
[13] A Giret and V Botti, Analysis and design of holonic manufacturing systems. 18th International Conference on Production Research (ICPR2005) 2005.
[14] Y Gufflet and Y Demazeau, Applying the paco paradigm to a three-dimensional artistic creation. 5th International Workshop on Agent-Based Simulation, ABS'04 Lisbon, Portugal, 2004, SCS, pp. 121–126.
[15] K Hallenborg, A J Jensen and Y Demazeau, Reactive agent mechanisms for manufacturing process control. proceedings of The 3rd International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS-2007) 2007, pp. 399–403.
[16] B Horling, V Lesser, R Vincent, T Wagner, A Raja, S Zhang, K Decker and A Garvey, The TAEMS White Paper, 1999.
[17] I IGN, Generalisation modelling using an agent paradigm, Tech. Rep. AG-98-07, AGENT, 1998.
[18] G Maionea and D Naso, A soft computing approach for task

contracting in multi-agent manufacturing control. Computers in Industry, Vol. 52, No. 3, 1996, pp. 199–219.
[19] M Wooldridge and N R Jennings, Intelligent agents: Theory and practice. Knowledge Engineering Review, Vol. 10, No. 2, 1995, pp. 115–152.