Applying eco problem solving to the control of an adaptative manufacturing cell
Christophe Sohier, Bruno Denis, Pierre Bourdet

To cite this version:
Christophe Sohier, Bruno Denis, Pierre Bourdet. Applying eco problem solving to the control of an adaptative manufacturing cell. IMACS-IEEE Multiconference Computational Engineering in Systems Applications (CESA’96), Jul 1996, Lille, France. hal-03745219

HAL Id: hal-03745219
https://hal.science/hal-03745219
Submitted on 3 Aug 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Applying eco problem solving to the control of an adaptative manufacturing cell

Christophe SOHIER, Bruno DENIS, Pierre BOURDET

Laboratoire Universitaire de Recherche en Production Automatisée
Ecole Normale Supérieure de Cachan
61, Av. du Pdt. Wilson
94235 Cachan cedex

ABSTRACT

The research of adaptivity and reactivity in the control of adaptative manufacturing cells implies distribution of intelligence in decision making. It also entails the difficulty to master and optimize real-time decision-making. The aim of this paper is to study the interest of multi-agent system and eco-resolution, in the control of adaptative manufacturing cells.

After an introduction of the problem we deal with, we present principles of the eco-resolution introduced in the field of distributed artificial intelligence. Secondly, we show how model a cell controller with eco-agents acting in an eco-problem. Then, we propose some extensions to the eco-resolution in order to suit to the control of a physical system of production. A precise model of a generic eco-agent is presented in the third session. Lastly, milling cell controller developed at LURPA is studied as example.

1. INTRODUCTION

The economic environment of subcontracting workshops and workshops for prototypes, leads to manage a great variety of products and to minimize unproductive times. These constraints involve a flexible command of manufacturing systems, to handle product diversity, events reactions and to get an easy usage for systems [1] [2].

In this type of manufacturing system, manipulated products are discreet (parts, tools and equipment). For small batch productions or production of prototypes, tools and equipment flows are not insignificant regarding part flows. Then, tools and equipment flows ought to be carefully considered.

A representative physical structure of manufacturing systems is a structure made up of cells (set up cells, machining cells, measuring cells) and a logistic system (transport and warehouse system). The hierarchical control system is the most suitable for studied systems [3] [4]. It allows to distribute decision-making as well as reaction to local events. The highest level has a partial view and takes decisions on objectives to short and mean-term.

For the experimental workshop of LURPA, we have retained a three levels architecture: workshop, cell and equipment levels. In this article, we deal with the cell level. At this level, it is necessary to control the setting-up of productions (putting part container, fixtures, tools and so on, into cell), to control the process of productions (clamping parts in fixtures, removing parts from fixtures, machining parts and so on), and the removing of productions (removing of parts, part containers, fixtures, tools and so on).

The cell controller has to schedule operations belonging to setting-up, processing and removing of production. A lot of these operations can be executed in a concurrent manner, but they often share the same resources in the cell. To obtain the best performances of the cell, the current state of resources must be take into account to schedule [5]. Thus, the organization can not be decided during the design of cell controller, neither at the moment when the method of production of new product is planned. At each moment, the scheduling of the cell has to be elaborate by itself, according to the current state of the cell. To do that, a functional view of the cell has to be provided to the cell controller, and not a view of the organization of productions.

In order to design adaptative cell controllers, we have used techniques from the field of distributed artificial intelligence and we apply them on the area of the FMS control [7] [8]. These techniques are multi-agent systems and eco-resolution. On such a system, the whole behavior is distributed into a set of eco-agents. During the processing, the global behavior is the result of the eco-resolution, and the system is self planned.

In the next session, we describe eco-resolution principles and how apply it in the design cell controller. Then we propose a generic model of an eco-agent according to the cell controller requirements [6]. Lastly, an example provide.

2. PRESENTATION OF ECO-RESOLUTION

Eco-resolution principles

The eco-resolution is a method to solve problems based on a multi-agent system. A multi-agent system is a space where eco-agents act in interaction. Each eco-agent responds to autonomy and locality principles and is linked with relationships that it has established with others eco-agents. According to our requirements for a manufacturing cell controller, we choose agents' model based on the actors' model. Eco-agents act in a concurrent manner. They receive messages and act accordingly with their behaviors.

Each eco-agent has a goal, and it tries to reach it with elementary actions that it knows. To reach its goal an agent reacts according to mechanisms based on simple social behaviors: satisfaction, aggression and flee. For an eco-agent, the will to be satisfied is the will to reach its goal. Asking to an eco-agent to get satisfied is asking to reach its goal. Each eco-agent has a goal, and it tries to reach it with elementary actions that it knows. To reach its goal an agent reacts accordingly with their behaviors.

For the experimental workshop of LURPA, we have retained a three levels architecture: workshop, cell and equipment levels. In this article, we deal with the cell level. At this level, it is necessary to control the setting-up of productions (putting part container, fixtures, tools and so on, into cell), to control the process of productions (clamping parts in fixtures, removing parts from fixtures, machining parts and so on), and the removing of productions (removing of parts, part containers, fixtures, tools and so on).
some others that will be called “bolts”. To the opposite, an eco-agent can be aggressed. In this case, it has the obligation to flee. Fleeing is the answer to an aggression. Fleeing makes an eco-agent change his location in the environment to avoid conflict with other eco-agent’s wills [9][10].

Each eco-agent is characterized by one of these three internal states. An eco-agent can be satisfied (its goal is reached), searching satisfaction or escape (it flees because of an aggression).

A relationship of dependence between eco-agents is defined. This relationship defines an order of satisfaction of eco-agents. An eco-agent will satisfy itself only if eco-agents whose it depends are satisfied. This relationship allows to organize an eco-problem, and adds a new internal state to eco-agents: waiting for dependencies.

To solve a problem with principles of the éco-resolution, it is necessary to distribute the global goal into eco-agents. Each eco-agent will have to solve a part of the problem by reaching its goal. Mechanisms of aggression, flee and satisfaction are going to organize the multi-agent system. The problem is solving without describe the manner to solve it.

### Modeling a cell controller as an eco-problem

The aim of the cell controller can summarize to:

- move parts, tools, equipment and numerical control part programs, according to production requirement, and according to the state of resources,
- act on parts, tools, equipment and numerical control part programs (for example clamp, manufacture, execute…).

Each entity that moves in the cell, is modeled by an eco-agent. Its goal is to reach the location where the entity is supposed to be during the production. If a necessary resource required to move the entity is engaged, then its eco-agent will attack (aggression) the eco-agents using the resource (bolts) [11]. An eco-agent knows the different locations where it can stand, and actions that allow it to go from one of these locations to an other one. Dependence relationships are used to prevent an eco-agent to act while an other eco-agent has not reached its location. For example, if the will of a part is to go on the carousel, then the part depends on the fixture. The part waits the fixture on the carousel to research its own satisfaction (fig. 1).

### Eco-resolution extensions

Production cells can manage simultaneously several productions. That implies the presence of several eco-problems in a multi-agent system. Therefore, some eco-agent are in several eco-problems. We introduce the static eco-agent notion to describe permanent resources of cells (robots, carousels, machines, pick up and deposit stations, …) and the dynamic eco-agent notion to describe specific products to each production (parts, part containers, fixtures, numerical control part programs, …). Static eco-agents are designed once during the idea of the cell. For each design of a new production, new dynamic eco-agents are designed. Dynamic eco-agents of a production know themselves, and know static eco-agents, but they do not have to know dynamic eco-agents of an other production. That allows to have autonomous productions.

Productions are usually structured in three parts: setting-up, process, and removing. It is interesting to split an eco-problem on several eco-problems. To manage eco-problems, eco-planners are introduced. The goal of an eco-planner is to solve its eco-problem, and it reach its goal when its eco-problem is solved.

We have summarized a cell controller with two typical roles: move entities and act on entities. Already described eco-agents allow to move entities in the cell. To act on entities (i.e. on their eco-agents) we introduce eco-operators. The goal of an eco-operator is to modify attribute of an eco-agent. For example, an eco-operator can make execute a numerical control part program by changing the goal of the eco-agent “numerical controller”.

An eco-problem can be represented with a graph to show its structure (fig. 2).

### 3. INTERNAL MODEL OF AN ECO-AGENT

We have decomposed an eco-agent in four functional parts:

- an interface module, composed of a generic part, that allows to emit and to receive necessary generic messages for mechanisms of eco-resolution (Sender fled, Sender aggresses Receiver, Sender is satisfied, and Sender isn’t satisfied). A specific part allows others eco-agents to access to specific attributes. If the eco-agent offers services to the other eco-agents, this interface is equally used
- a module of behavior management (generic) that manages the internal behavior of the eco-agent, i.e. satisfaction, aggression and escape behaviors.
- a module of attribute management (specific) that allows with the specific part of interface module to offer a description of the eco-agent and eco-agent’s services. This part that has a description of the eco-agent environment.
- a module of management of actions (specific) that allows the agent to move in its environment. This module performs actions associated to specific satisfaction and escape behaviors. All actions are entirely defined during the agent design.

Eco-agents communicate by sending signed messages. Four kinds of messages are defined to insure the dialogue between eco-agent generic parts. These messages are:

![Fig. 1: Example of dependence relationships](image1)

![Fig. 2: Structure of an eco-problem](image2)
- Sender fled: an eco-agent receives this message when a bolt eco-agent (the sender) has escaped.
- Sender aggresses receiver: an eco-agent receives this message when it is a bolt for the sender, it has the obligation to escape.
- Sender isn’t satisfied: an eco-agent receives this message when the eco-agent it depends on is no longer satisfied, and it emits this message to eco-agents that depend on it when it is not satisfied.
- Sender is satisfied: an eco-agent receives this message when the eco-agent it depends on is satisfied, and it emits this message to eco-agents that depend on it when it is satisfied.

These messages are illustrated by the following example (fig. 3). This example is representative of all generic messages exchanged between eco-agents of a same eco-problem.

Fig. 3 : Generic messages exchanged between eco-agents

Relationships that allow to exchange messages between eco-agents of the example are following:
- eco-agent “fixture A” is the goal of eco-agent “part”
- eco-agent “carousel” is the goal of eco-agent “fixture A”
- eco-agent “fixture A” is a bolt for eco-agent “fixture B”
- eco-agent “fixture C” is a bolt for eco-agent “fixture A”

The generic module “behavior management” is going to be described. To model, both, the state of the module and its trades of messages with the other modules, we choose the SA-RT method [12].

On the context flow diagram “Manage eco-agent internal behavior” (fig. 4) the three other modules appear as external entities. All messages exchanged between “Manage eco-agent internal behavior” and “Interface manager” are generic messages.

The diagram “Manage eco-agent internal behavior” (fig. 5) is decomposed in seven process. The seven process are divided in three groups. The first group (1, 2 and 3) deals with actions (to escape or to be satisfy).

**Process 1: Manage_actions_demands**

**description**
This process manage all action requests from the agent. So, all internal processes send their action requests to this process, and outside, the action manager talk with only one internal process (this one). This, allows a good encapsulation.

end pspec

End of process 1
Process 5: Manage_aggressors

description

IF "X aggresses A"
    add X aggressor to a aggressors list
    IF A isn't fleeing
        send "aggress"
        keep on mind that A is fleeing
    END IF
END IF

IF "fled" (e.g. A flee is completed) FOR EACH aggressor in aggressors list
    send "A fled"
    empty the aggressors list
    keep on mind that now A isn't fleeing
END FOR

ENDIF

end pspec

The third group (6, 7) deals with dependence relationship.

Process 6: Watch_on_the_ones_A_depends_on

description

IF "X is satisfied"
    remove X to the 'non satisfied agent A depends on' list
    IF 'non satisfied agent A depends on' list is empty
        send "dependency OK"
    END IF
END IF

IF "X isn't satisfied"
    IF 'non satisfied agent A depends on' list is empty
        send "dependency NOK"
    END IF
    add X to the 'non satisfied agent A depends on' list
END IF

end pspec

Process 7: Inform_the_ones_that_depend_on_A

description

IF "satisfaction"
    FOR EACH ones that depend on A
        send "A is satisfied"
    END FOR
END IF

IF "no satisfaction"
    FOR EACH ones that depend on A
        send "A isn't satisfied"
    END FOR
END IF

end pspec

The finite state diagram “Control specification of Manage eco-agent internal behavior” (fig. 6) models the four internal eco-agent states: "satisfaction searching", "fleeing", "satisfaction", and the fourth coming from dependence relationships “waiting”. The “fleeing” state is split into “fleeing from wait” state and “fleeing from satisfaction” state because actions are different when the process leaves these states.

Process "Reach_the_goal_(satisfy)" (fig. 7) manages aggressions and actions to satisfy eco-agent.

Process 2.1: Identify_needed_action_to_be_satisfy

description

Find action to reach goal from the current location
Send "action to satisfy" and "locations which must be without bolts"

end pspec

Process 2.2: Identify_bolts

description

FOR EACH "locations which must be without bolts"
    read "location status"
    IF a bolt is present
        add this bolt to the list of “bolts”
    END IF
END EACH

wait list of "bolts" empty
send "no bolts for reach the goal"

end pspec

Process 2.3: Apply_satisfaction_policy

description

When "no bolts for reach the goal" and “action to satisfy” then
Send action to satisfy
When "action i completed"
Send "goal completed"

end pspec

Process “Answer_to_an_aggression_(flee)” (fig. 8) has a same structure as “Reach_the_goal_(satisfy)”. Its process specification will not develop any more in this paper.
4. EXAMPLE: MILLING CELL CONTROLLER

This example is based on one of production cells of experimental workshop of LURPA. It concerns a milling cell that is constituted following components:
- a milling machine, with a Numerical Controller,
- a 5 axis robot, with a Numerical Controller,
- a carousel and three pick up and deposit stations connected to the workshop logistic system, this set of equipment is managed a Programmable Logical Controller,
- a unix computer, as cell controller, which is in charge of scheduling basic actions of previous components, in order to get a coherent behavior of the cell.

The problem to solve, is some of a production. At the beginning of the problem, a part is on a container outside the cell, and a corresponding fixture is outside the cell too. At the end of the problem, the container has to be on the station, the part has to be in the fixture, and the fixture has to be on the carousel (fig. 9).

Now, to allow eco-agents to act in their environment, actions are developed. Each actions lets agent go from a location to an other one, and lets it know all its possible bolts (tab. 1) (fig. 10).

To finish the eco-problem description, dependencies have to be set up. Part eco-agent depends on the part container eco-agent and the fixture eco-agent satisfactions. Part eco-agent will try to reach its goal when the part container eco-agent and the fixture eco-agent reach their goals, but not before. This eco-problem is presented fig. 11, it is a part of the multi agent system, limited with dashed oval. The bottom of the figure shows static eco-agents already design since the cell controller design. On the right side of the figure, other eco-problems appear, that means other productions exist on the cell.
dependence relationships between eco-agents impose the order of dynamic eco-agent satisfaction. Only “fixture” and “part containers” eco-agents can try to reach satisfaction. They apply their behavior of satisfaction. The “parts container” eco-agent has no bolt, it start the action to go on “pick up and deposit station”. On the other hand the “fixture” eco-agent has a bolt because “carousel” is engaged. It aggresses the “fixture” eco-agent that prevents it to satisfy. When the bolt eco-agent has escaped, it informs its escape by sending the message “Sender fled” to the aggressor agent. The “fixture” eco-agent has no longer bolt, therefore it can satisfy, i.e. it can go to “carousel”. When it is satisfied, it informs ones that depend on its satisfaction state by sending the “sender is satisfied” message. Now, the “part” eco-agent only depends on the “fixture” eco-agent that has reached its satisfaction. It can reach its situation. All eco-agents of the eco-problem are satisfied, then the eco-problem is solved.

This example of resolution shows the succession of actions to solve the problem from an initial situation. From a different initial situation, another succession of actions would have been found to solve the same eco-problem.

5. CONCLUSIONS

We proposed an extended eco-problem to model adaptative cell controllers. Static eco-agents allow us to model cell dependant entities, and dynamic eco-agents allow us to model production dependant entities. Eco-planner and eco-operator have been introduced to describe production structure.

The interest of the eco-agent modelling, is to separate the functional description of a manufacturing production from its actual organization. Then, add a new production in a cells consist of create a new eco-problem. No new schedule is required. Eco-agent behavior will manage the new production according to the current state of the cell.

SA-RT method has been used to produce a generic model of the behavior of eco-agents. Our current works deal with eco-agent implementation with Ada language. Each eco-agent is implemented in a Ada task, and language genericity properties are used to implement generic part of each agent’s module.