TOWARDS ROBUST HYBRID CENTRAL/SELF-ORGANIZING MULTI-AGENT SYSTEMS

Yaser Chaaban, Jörg Hähner and Christian Müller-Schloer
Institute of Systems Engineering, Leibniz Universität Hannover, Appel Str.4, Hannover, Germany

Keywords: Organic computing, Self-organisation, Coordination, Autonomous vehicles, Robustness, Multi-agent systems, Artificial intelligence.

Abstract: The Organic Computing initiative uses life-like properties such as self-organisation, self-optimisation and self-configuration towards building today's technical systems as flexible, robust, and adaptive systems. In a previous paper, we proposed a system for coordinating semi-autonomous agents under the framework of Organic Computing. It uses abstractions of observer and controller to add robustness and solve scheduling/allocation problems. In this context, the path planning and the observation of the agents were presented and also the detection of deviations in different situations was discussed. In this paper, we introduce control features of the system designed to deal with these types of deviations. That leads in turn to intervene in time when it is necessary, so that the system remains demonstrating robustness. Furthermore, this paper addresses the conflict between a central planning algorithm and the autonomy of the agents. A hybrid central/self-organizing multi-agent system is introduced solving this conflict.

1 INTRODUCTION

The behavioural intelligence can be seen as a mixture of flexibility, robustness and adaptiveness of behaviour. This mixture is however the key idea of developing today's technical systems which use the Organic Computing (OC) concept. The Organic Computing initiative uses life-like properties such as self-organisation, self-optimisation and self-configuration towards building those systems as flexible, robust, and adaptive systems.

Robust system shall behave or act appropriately according to situational needs. But this is not guaranteed in novel systems which have their complexity and whose environment is changing dynamically, because that leads to unexpected system behaviour.

Because environments of complex systems may change dynamically, self-organising systems should be provided with some degrees of autonomy so that they can adapt their behaviour to new environmental situations. This autonomy as well as errors, disturbances and deviations may cause an unwanted emergent behaviour (Mnif et al., 2007). Therefore, the system should be observed (e.g., by an Observer) and controlled (e.g., by a Controller) so that this emergent behaviour can be prevented and the system performance remains effective as long as possible. A generic Observer/Controller (O/C) architecture has been proposed in (Richter et al., 2006) in order to establish the controlled self-organisation in technical systems. Figure 1 shows a generic o/c architecture. As depicted in Figure 1 the behaviour of the technical system can be evaluated to intervene in time when it is necessary. The o/c architecture has a set of sensors and actuators to measure system variables and influence the system. The observer observes the system state and its dynamics, quantifies them and aggregates its observations as a vector of situation parameters. These parameters are then sent to the controller. The controller evaluates the situation parame-

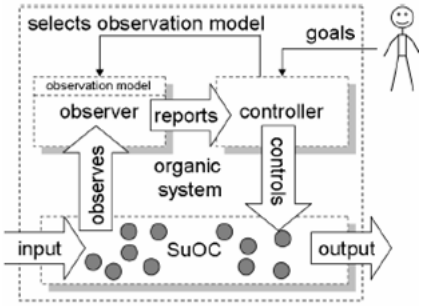

Figure 1: Observer/Controller architecture.

Chaaban Y., Hähner J. and Müller-Schloer C. (2010).
TOWARDS ROBUST HYBRID CENTRAL/SELF-ORGANIZING MULTI-AGENT SYSTEMS.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 341-346
DOI: 10.5220/0002761003410346

ters and influences the system under observation and control (SuOC) with respect to the given goal by the user. In previous paper, we proposed a system for coordinating cars at intersections using an o/c architecture (Chaaban et al., 2009). The traffic intersection is regulated by a controller, instead of having a traffic light. The cars send messages (requests) to the intersection (path planning unit of the controller) and then get appropriate trajectories. These trajectories guarantee a coordinated behaviour with the other cars in order to avoid traffic jams in the centre of the intersection. Figure 2 shows a screenshot from our project. In that earlier paper, we focused on the path planning

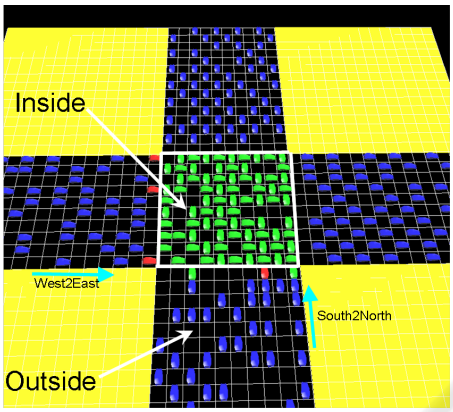

Figure 2: The intersection without traffic lights.

and the observation of the agents and also the detection of deviations in different situations.

In this paper, we describe the control process of the system designed to deal with these types of deviations. That leads in turn to intervene in time when it is necessary, so that the system remains demonstrating robustness. This control process depends on our hybrid central/self-organizing concept, which solves the conflict between a central planning algorithm (path planning in the traffic intersection) and the autonomy (deviations from the plan) of the agents.

The scenario of traffic intersection offers a lot of cases that can be used to verify the new investigated concept. In this scenario, a resource sharing problem (resource sharing conflict) arises which has to be resolved in order to avoid collisions in the centre of the intersection (a shared resource). In this context, the hybrid central/self-organizing concept is used to solve this coordination problem. It aims to keep the intersection robust when deviations occur in the behaviour of the cars, so that the cars can move reliably in their environment.

2 THE ORIGINAL SYSTEM

Previously, we proposed a new multi-agent approach which deals with the problem occurring in the system wherever multiple agents (cars) move in a common environment (intersection without traffic lights). We presented the desired system architecture together with the technique that is to be used to cope with this problem. This architecture was an o/c architecture adapted to the scenario of traffic intersection.

The system under observation (cars within the centre of the intersection) is considered as a set of elements possessing certain attributes in terms of multi-agent systems. This means that every car in the system is an agent. Every car by itself is assumed to be egoistic (because the driver here is autonomous and he tries quickly to cross the intersection and perhaps he does not obey his trajectory). Therefore competition situations arise due to the egoistic behaviour (competition-based behaviour) of cars, which in turn leads to a traffic jam in the centre of the intersection.

2.1 Path Planning

Path planning delivers collision-free trajectories for all cars. Path planning has to be done only for cars inside the centre of the intersection. A car outside the centre of the intersection has only local rules, through which this car tries to move forward avoiding collisions with other cars.

When a car arrives at a border of the centre of the intersection, it sends a message (request) to the intersection (path planning unit of the controller). This message has to be responded to by the intersection controller so that the car is being able to cross the centre of the intersection (shared resource) safely, if no unexpected errors occur within this process. The path planning takes into consideration other cars and the geometry of the intersection in the configuration space-time. It calculates an appropriate trajectory and sends it to that car which would be entering the centre of the intersection. Furthermore, the calculated trajectory is stored in the memory of the trajectories. The enquiring car gets its trajectory which guarantees a coordinated behaviour with the other cars in order to avoid traffic jams in the centre of the intersection. We assume that every car obeys its trajectory. But this is not guaranteed. Therefore the observer of the intersection observes whether the current travelled path of a car in the centre of the intersection corresponds to the planned trajectory of this car in the memory of the trajectories. If this is not the case, then the intersection controller is informed so that it could intervene in time if it would be important.

The A*-procedure for path planning of cars is applied in three dimensional configuration time-space. It makes an independent planning of the paths for the individual agent (cars) in their configuration time-spaces which extends the configuration space of the agent by a time axis.

2.2 Observation

The observer concentrates at present only on the intersection. Therefore, other observers in order to observe the agents (cars) on the way are not considered.

In the centre of the intersection every car has to obey its trajectory as planned. Since deviations from the planned trajectories are possible, the monitoring is done in order to detect the deviations and to intervene dynamically through re-plan trajectories of the affected cars. The observer of the intersection aggregates its observations as a vector of situation parameters. These parameters are then sent to the controller.

2.3 Detection of Deviation

The observer compares the two states (should-be and actual states) of every car in order to detect whether any deviation from the plan occurred. When the observer detects any deviation, then it has to find the deviation class. The possible classes of deviations, which could be detected through the observer in this system, are: 1- Accident. 2- Autonomy. 3- Accident and Autonomy (Chaaban et al., 2009).

3 THE HYBRID CENTRAL/SELF-ORGANIZING CONCEPT

The generic o/c architecture has to be customised to different scenarios. The distribution possibilities of the proposed architecture are varying from fully central to fully distributed architecture (Branke et al., 2006). The three main options to realize the generic architecture as depicted in the Figure. 3 are:
(a) Central: One o/c for the whole system.
(b) Decentral: One o/c for each subsystem.
(c) Multi-level: One o/c for each subsystem as well as one (or more) for the whole system.

In this paper, we are introducing the term, hybrid central/self-organizing multi-agent system. It is a new possibility of the distribution of the proposed architecture. The new architecture is a special form of the fully central architecture, in which some autonomous agents can leave the control of the fully central architecture and also to behave in fully autonomous way. Figure 4 shows the main idea of this hybrid central/self-organizing concept arising from the fully central architecture.

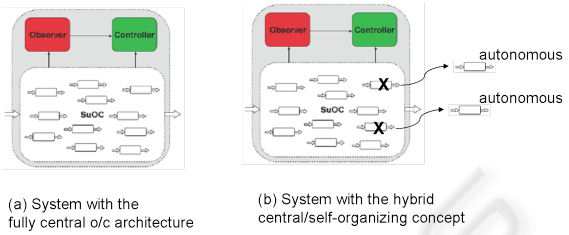

Figure 4: The hybrid central/self-organizing concept.

Organic Computing searches for concepts to achieve controlled self-organisation as a new design paradigm, which is necessary to cope with degrees of freedom required by the process of self-organisation (Cakar et al., 2007). The choice of the appropriate o/c realization is a design decision that has to be done by the developer in the design phase of the technical system. In this work, the hybrid central/self-organizing concept aims to increase the autonomy of agents in the central architecture. This means, our hybrid concept tolerates that some agents behave in fully autonomous way in the central architecture. It solves the conflict between a central planning algorithm and the autonomy of the agents. Here, the autonomy is recognized as a deviation from the plan of the central algorithm, if the agents are not respecting this plan. Consequently, our new concept comprises the use of a central o/c architecture, autonomous agents and deviations from a central plan in order to solve coordination problems in multi-agent systems. Additionally, it keeps the system robust when deviations occur in the system behaviour, so that the agents of a system can move reliably in their environment.

3.1 Decision Making

This paper focuses on the control process of the system to deal with the occurred deviations.

The decision maker is the central part of the controller. The controller uses the decision maker to take a decision how it can intervene most suitable when it is necessary so that the system can be influenced with respect to the given goal by the user. The given goal of the user in the introduced scenario of this work is to keep the system demonstrating robustness in spite of emergent behaviour which could be appeared in the system, so that the agents (cars) can move reliably in their environment in order to cross over it (intersection) quickly as soon as possible. That would be done

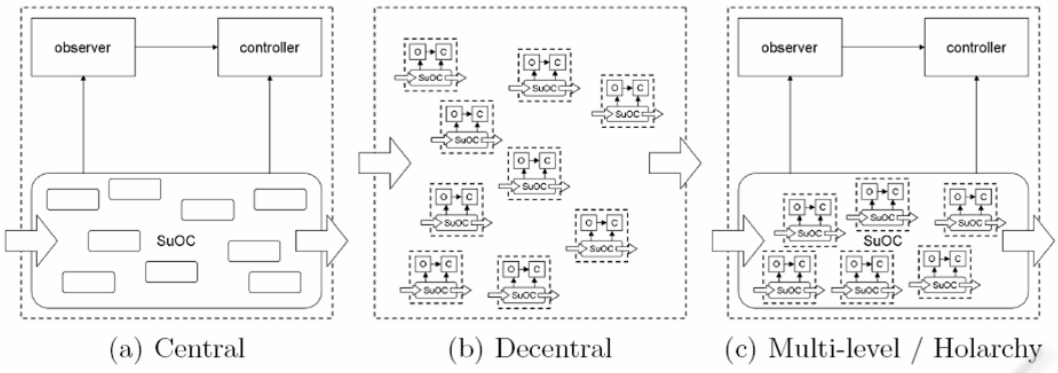

Figure 3: Distribution possibilities of the generic observer/controller architecture.

in addition to get autonomous traffic as possible with low delays. The decision maker is activated when the controller gets the situation parameters which contain a deviation message. On the other side, when there is no deviation, this means that everything is as planned and the decision maker will not be used here.

Algorithm 3.1. Overview of the controller algorithm with the aid of the observer.

The observer calculates and checks δ:
$\delta \leftarrow$ (should-be state) XOR (actual state)
if ($\delta = 0$) **then**
- there is no deviation, and everything is as planned, and the decision maker will not be used here.

else
while ($\delta \neq 0$) **do**
- there is a deviation from the plan.
- increase the counter of the Autonomy Detector: (AD = AD + 1).
if (AD $\geq$ the threshold of the emergency) **then**
activate the state of emergency
end if
- the observer finds the deviation class, and send a deviation message to the controller.
- the controller reads the deviation message and re-plans the trajectories of the affected agents (cars), and send it to the system.
- wait for the next simulation tick.
- the observer calculates again and checks δ: $\delta \leftarrow$ (should-be state) XOR (actual state).
end while
end if

3.2 The Controller Algorithm

See Algorithm 3.1. for an overview of how the controller algorithm works and cooperates with the observer. This algorithm allows the controller to intervene dynamically through re-plan trajectories of the affected agents (cars), when the observer has detected deviations from the planned trajectories. Future work includes further investigation on intervention tasks. The simulation parameter AD (Autonomy Detector) plays a major role in the decision process. AD represents the degree of the system sensitivity to the deviations that occur. It can be adjusted to a certain value according to the used scenario. E.G., when the scenario is risky (e.g. cars scenario), the AD can be adjusted to a very low value. On the other side, the AD can be adjusted to a higher value. When the value of AD reaches the adjusted value (the threshold of the emergency), the controller activates the emergency state and also stops all agents (cars) and does not give any new trajectories to the other agents.

4 PERFORMANCE EVALUATION

In this section, we present an initial evaluation of our system using the model of a traffic intersection, which was designed and described in our earlier paper (Chaaban et al., 2009). Due to space limitations, we include only our main result. In future work, we intend to present a more complete empirical evaluation including experiments with other metrics for estimating the overall reduction of the Performance (throughput) of the system, in which deviations from the plan of the controller occur.

4.1 System Performance Metrics

In our previous work (Chaaban et al., 2009), we have measured the two fundamental performance metrics in traffic engineering: the throughput and the latency (the waiting time), because a high throughput and a low latency are always needed in traffic engineering.

Throughput is the total amount of cars that leaved the intersection (simulation area) over time, whereas the mean latency is the mean waiting time (ticks) needed by cars to traverse the intersection.

In this paper, another metric is measured, the response time. Response time is an important metric for real time systems, because in these systems short response time is required. This short response time is required in this work, because cars approaching the intersection need trajectories as soon as possible.

Response time is the time which takes a system to react to a given input. The response time here is the time between the moment when the path planning unit in the controller of the o/c architecture gets messages (requests) from the system (cars) and the moment when it sends appropriate trajectories to the system (cars). Thus, currently the average response time is the average used computation time of the search for the best appropriate trajectories of cars. In this scenario, the system with the o/c architecture will provide a better system performance if the response time is shorter.

4.2 Evaluation Scenarios

We used four different test scenarios to measure and compare the system performance. These scenarios result from the change of values of the following two simulation parameters. The first simulation parameter is the maximum number of cars in each direction. The second simulation parameter is the production rate of cars in each direction (Traffic Level). The four different used evaluation scenarios ensure that the system performance in various combinations of the parameter remains effectively. We called this four test scenarios: (Equal-Equal), (Equal-Not Equal), (Not Equal-Equal) and (Not Equal-Not Equal) (Chaaban et al., 2009).

4.3 First Results

Currently, the path planning of the o/c architecture only has been implemented in order to deliver collision-free trajectories for all cars. Thus, the results described here are measured assuming that no deviations occur in the system. Recalling, that one tick in the simulator means one time step.

Additionally, we have implemented the reservation algorithm for the trajectories of cars in two ways trying to get better response time of the system. The first way is **"AllTrajectoriesVector"**. Here, every cell in the intersection is an object (instance) of the class SpaceTimePoint (x,y,time). Each trajectory in turn is a vector. This vector contains all points (SpaceTimePoints) which represent a trajectory. Accordingly, the AllTrajectoriesVector is a vector that contains all trajectories of cars. The second way is **"PhotoOfGrid"**. Here, for each tick (each unit of time) in the simulation a photo for the whole area of the intersection will be stored. Therefore, in every cell an "AgentID" (CarID) is saved if this cell at this time (at this tick) for this agent (car) is reserved. Each level represents a photo of a specific tick of the simulation. Thus, a photo represents the coordinates (x,y), whereas a level represents the third axis (time), so that the configuration time-space is formed. Each photo is implemented as a HashMap, where the keys are the ticks and the values are the photos:

$$(key, value) = (Tick, PhotoOfGrid) \qquad (1)$$

Figure 5 shows the structure of the PhotoOfGrid-way.

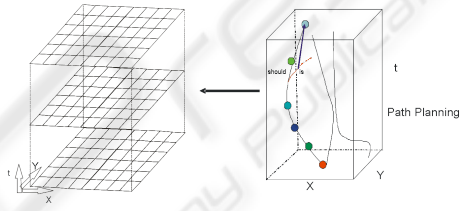

Figure 5: The structure of the PhotoOfGrid-way.

We have measured the average response time of the system for the two reservation ways after 3000 ticks in two selected scenarios. The scenario I is a simple scenario in terms of a small number of cars. We have measured in scenario I the average response time of the system in the case that traffic level of cars in south-north and west-east directions is only (1) cars/tick, whereas the maximum number of cars in each direction is only (20) cars. The scenario II is a complex scenario in terms of a large number of cars. We have measured in scenario II the average response time of the system in the case that traffic level of cars in south-north and west-east directions is (4) cars/tick, whereas the maximum number of cars in each direction is (100) cars.

Table 1 shows the resulting average response times in the two reservation ways.

Table 1: Average response times in (ms).

	Scenario I (simple case)	Scenario II (complex case)
AllTrajectoriesVector	0.167	0.930
PhotoOfGrid	1.062	12.447

We can note that the second reservation way (PhotoOfGrid) requires about (6) times longer time than the first reservation way (AllTrajectoriesVector) in the scenario I, whereas it is about (13) times longer in the

scenario II. That means, the reservation way (AllTrajectoriesVector) has approximately a quadratic complexity, whereas the reservation way (PhotoOfGrid) has approximately a cubic complexity, because time is additional to the (x,y) form this 3-D configuration. Since the reservation way (AllTrajectoriesVector) outperforms significantly the other reservation way (PhotoOfGrid) by computation time in several situations, we have further measured only the first way (AllTrajectoriesVector).

Figure 6 shows the system performance (average response time) for the scenario I (Equal-Equal) after 3000 ticks using the reservation method (AllTrajectoriesVector). We have measured in this scenario the

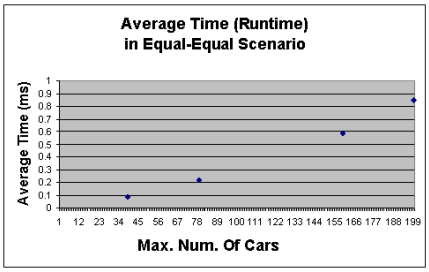

Figure 6: The average response time of system in scenario I (Equal-Equal) after 3000 ticks using the reservation method (AllTrajectoriesVector).

average response time of the system in the case that the traffic level of cars in south-north and west-east directions is (5) cars/tick, whereas we have repeated the measurement in the cases that the maximum number of cars in each direction is 20, 40, 80 and 100 cars. Note here, on the x-axis is the total amount of cars in the two directions together. The value of the average response time of the system in this scenario is about (0.1) ms when the total number of cars in the two direction is (40) cars, whereas about (0.83) ms by (200) cars. This means, that the average response time of the system increases approximately quadratically requiring less that (1) ms when the total number of cars in the two direction is (200) cars.

5 CONCLUSIONS

The choice of the appropriate o/c realization is a design decision that has to be done by the developer in the design phase of the technical system. In this paper, we presented a new approach towards building a robust hybrid central/self-organizing multi-agent system. This approach solves the conflict between a central planning algorithm and the autonomy of the agents. Additionally, it aims to increase the autonomy of agents in the fully central architecture. This means, our hybrid concept tolerates that some agents behave in fully autonomous way in the central o/c architecture. The scenario used in this paper is a traffic intersection without traffic lights. We introduced control features of the system designed to deal with deviations from the plan which can occur in the system behaviour. These control features intend to intervene in due time when it is necessary so that the system remains demonstrating robustness. Finally, as evaluation metric to measure the system performance, we used the response time in four different test scenarios in various combinations of parameters.

6 FUTURE WORK

Since we implemented the generic o/c architecture adapted to our traffic scenario and accomplished our experiments assuming that no deviations occur in the system, the next step is to continue with the implementation of the case when deviations occur in the system to completely realize our vision. Then, we will measure the system performance and compare the two cases, the system performance with and without deviations. This comparison will be used to determine whether the system performance remains effective as long as possible when a deviation occurs and consequently to assure a robust system.

REFERENCES

Branke, J., Mnif, M., Müller-Schloer, C., Prothmann, H., Richter, U., Rochner, F., and Schmeck, H. (2006). Organic computing - addressing complexity by controlled self-organization. In *ISoLA*, pages 185–191.

Cakar, E., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. (2007). Towards a quantitative notion of self-organisation. In *IEEE Congress on Evolutionary Computation*.

Chaaban, Y., Hähner, J., and Müller-Schloer, C. (2009). Towards fault-tolerant robust self-organizing multi-agent systems in intersections without traffic lights. In *Cognitive09: Proceedings of The First International Conference on Advanced Cognitive Technologies and Applications*, November 15-20, 2009 - Athens, Greece. IEEE, (To appear).

Mnif, M., Richter, U., Branke, J., Schmeck, H., and Müller-Schloer, C. (2007). Measurement and control of self-organised behaviour in robot swarms. In *ARCS 2007*, volume 4415 of *LNCS*, pages 209–223. Springer.

Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., and Schmeck, H. (2006). Towards a generic observer/controller architecture for organic computing. In *INFORMATIK 2006*. Bonner Köllen Verlag.