Gradient Field-Based Task Assignment in an AGV Transportation System

Danny Weyns
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
danny@cs.kuleuven.be

Nelis Boucké*
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
nelis@cs.kuleuven.be

Tom Holvoet
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
tom@cs.kuleuven.be

ABSTRACT

Assigning tasks to agents is complex, especially in highly dynamic environments. Typical protocol-based approaches for task assignment such as Contract Net have proven their value, however, they may not be flexible enough to cope with continuously changing circumstances. In this paper we study and validate the feasibility of a field-based approach for task assignment in a complex problem domain.

In particular, we apply the field-based approach for task assignment in an AGV transportation system. In this approach, transports emit fields into the environment that attract idle AGVs. To avoid multiple AGVs driving towards the same transport, AGVs emit repulsive fields. AGVs combine received fields and follow the gradient of the combined fields, that guide them towards pick locations of transports. The AGVs continuously reconsider the situation of the environment and task assignment is delayed until the load is picked, improves the flexibility of the system.

Extensive experiments indicate that the field-based approach outperforms the standard Contract Net approach on various performance measures, such as the average wait time of transports and throughput. Limitations of the field-based approach are an unequal distribution of wait times across different transports and a small increase of bandwidth occupation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Design

1. INTRODUCTION

Multiagent systems (MASs) are generally considered as an approach to build complex software systems. Adopting an agent-oriented approach means decomposing a system into multiple, autonomous entities that can act and interact in a shared environment to achieve the system goals. A MAS provides a decentralized architectural solution to a software problem that aims for qualities such as robustness and flexibility.

The application that is central in this paper is an Automatic Guided Vehicle (AGV) transportation system. An AGV system is a fully automated industrial transport system that makes use of multiple AGVs that have to transport loads in an industrial environment, e.g. a warehouse. One way to control an AGV system is through a central server. This server maintains all relevant information, needed to decide which AGV has to do what and which route is to be taken. In a joint research project between AgentWise taskforce and Egemin, a producer of automated logistic systems, we investigate the feasibility of applying the MAS paradigm to an AGV transportation system [5, 21]. In this approach, the control is distributed over the different AGVs, who cooperate to achieve the goals of the system. One major challenge in an AGV transportation system is task assignment, i.e. the assignment of transports to AGVs. *Transports*, which consist of picking up a *load* on a location (i.e. the pick location) and transporting it to another location (the drop location), are generated by an external system. Transports are dynamically created when needed and the transport profile (i.e. the order and pace in which transports are created) is irregular and unpredictable. Due to the highly dynamic nature of transport creation, the assignment of transports to AGVs is complex. Egemin currently applies a policy that pre-stages the AGVs based on a fixed set of rules. This approach however, lacks flexibility to cope with the continuously changing circumstances in the environment.

Typical protocol-based approaches for task assignment such as Contract Net have proven their value, however, they may not be flexible enough to cope with continuous dynamics in the environment. The main goal of the work reported here is to examine the *feasibility* of using a *field-based approach* to achieve *task assignment* in a complex domain, in casu an AGV transportation system. We aim to investigate the advantages as well as the limitations of the approach in this real world setting. In the field-based task assignment for AGVs, transports emit fields that attract idle AGVs, while competing AGVs emit repulsive fields. Each idle AGV follows the gradient of the combined fields it receives, which guide the AGV towards the pick location of a transport. To validate the field-based approach, we implemented it in an AGV simulator [1]. In the simulations, we use data of a realistic AGV system, including a real-world map and transport profiles. We have performed various performance measures, such as the average time a transport has to wait before it is executed, the throughput of the system, and the number of messages that are transmitted. The test results are compared with a reference algorithm for task assignment that uses a Contract Net protocol [19]. In this paper, we give an overview of the field-based approach for task assignment, and we discuss the main

*Supported by Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

results obtained from this study. A detailed description of additional tests not included in this paper can be consulted in [17].

Overview. The remainder of this paper is structured as follows. In Section 2 we briefly introduce the AGV application. Section 3 discusses related work on task assignment and field-based techniques applied in MAS. In Section 4 we describe the approach of field-based task assignment. Section 5 discusses test results and evaluates the approach. Finally, we draw conclusions in Section 6.

2. THE AGV APPLICATION

An AGV system is a fully automated industrial transport system, that uses multiple AGVs to transport loads (raw materials, finished products, etc.) in an industrial environment such as a warehouse. In this section we give a brief overview of the AGV application. Subsequently the map of the environment, the transports, and the AGVs are described.

2.1 Map of the Environment

The AGVs are restricted to follow predefined paths in the environment, on which they navigate by using sensors and stationary beacons. These paths are defined in a *map* of the environment. A map is logical representation of the physical layout on the factory floor, it consists of a network of stations and connecting segments that are accessible by the AGVs. A station is a predefined point on the map, stations can hold special locations, such as a pick or drop location for transports, a parking place for AGVs, or a place where AGVs can reload their battery. A segment defines in which direction(s) AGVs can drive. Additional information may be associated to segments, e.g. speed limits, mutexes that indicate that only one AGV can enter a particulate zone at a time, etc. In section 5 we give an example of a realistic map that we have used for testing.

2.2 Transports

The goal of the AGV system is to let AGVs transport loads. A transport is created by an external system, typically a warehouse management system. A transport represents an order to move a load from a pick location to a certain drop location. Each transport has a priority from 1 to 10, with 10 being the most important. A transport's life-cycle starts when it enters the system. The transport is started when an AGV picks up the load and is terminated when that AGV drops the load at the drop location. The priority of a transport may increase over time, e.g. as a result of long waiting times.

2.3 AGVs

An AGV is a computer controlled, unmanned vehicle that is capable of transporting loads in an industrial environment. An AGV is capable of picking up loads, driving them around and dropping them. AGVs can communicate by means of a wireless LAN. An AGV navigates by storing an internal map of the environment and using internal and external sensors, in combination with stationary beacons. AGVs are equipped with low-level control software that handles the physical interaction with the environment such as staying on track, turning, or determining the current position.

To allow smooth and save movement, an AGV has to make routing decisions over a certain distance in advance. This distance, called the *look-ahead*, is measured according to the logical topology of the environment map. The *look-ahead buffer* associated with an AGV is an ordered set of stations and segments that holds the routing decisions, representing the path that the AGV will follow with certainty, unless something catastrophical (such as a crash) happens.

In this paper we focus on achieving *task assignment* by applying a field-based approach. During task assignment the AGV will use a simple A* algorithm [8] to *rout* towards a transport. Besides task assignment, other important concerns are collision avoidance and deadlock prevention, however, these concerns are not further discussed.

3. RELATED WORK

In this section, we discuss a number of approaches related to our field-based task assignment approach. First we discuss a number of general approaches for task assignment. Task assignment is an extensive domain of research. Here we mainly focus on well-known Contract Net-based mechanisms. Next, we discuss a number of related field-based techniques for MAS described in literature.

3.1 Task Assignment

Efficiently assigning tasks to individual entities in a decentralized system is a common, but certainly not a simple problem [7]. A classic way to achieve task assignment is through auction-based mechanisms, e.g. the widely known and extensively used *Contract Net* protocol (CNET [19]). CNET is based on the notion of call for bids on markets, where managers request for bids and bidders bid to perform the task. In a first step, the manager sends a description of the task to perform to all the bidders. In a second step, the bidders draw up a proposal based on the description of the task and send it to the manager. In a third step, after the manager has received a proposal from all bidders or after a deadline has expired, the manager evaluates the received proposals and assigns the best bidder the task.

In the original CNET protocol, a bidder that has placed a bid for a task (step 2), cannot place a bid for a new task before the first task has ended. In [10], an improvement to CNET is proposed that allows a bidder to place bids for multiple unassigned tasks. This is achieved by giving the bidder the option to refuse the task in step 3 of the original protocol. The manager will then award the contract to the second best bidder, who can again accept it or refuse it, etc.

The CNET protocol can also be extended to allow cascading task assignment, where the receiver of a task proposal starts one or more new CNET protocols to delegate the task or parts of the task to other agents [10]. For example, a system of holonic agents can be used to improve efficiency when dealing with complex tasks that require the collaboration of several agents [11]. A holonic agent consists of several agents but interacts with agents outside the holon as a single agent.

The CNET protocol suffers from low flexibility when dealing with tasks that can not immediately be started upon the allocation (delayed commencement of tasks [3]). Consider for example the CNET protocol applied to task assignment in an AGV system. When a new transport enters the system, a suitable AGV is selected through the CNET protocol and is assigned to the transport. Suppose that while the assigned AGV is driving towards the pick location of the transport, a new transport appears that is more suitable to the AGV (e.g. it is closer or has a higher priority). The AGV will not be able to execute the second transport, since it is already assigned to the original transport. Thus, the CNET protocol achieves an *immediate* and *permanent* assignment of tasks, making it impossible for the system to react flexibly and adaptively when dealing with tasks with delayed start in a highly dynamic environment. [2] provides more flexibility by assigning tasks in two phases (first phase temporarily assignment, next phase definitive assignment) but still does not provide full flexibility in case of delayed commencement.

Another interesting approach for task allocation is described in [6] that is based on the exchange of *tokens* among agents. The authors describe an example of robots that drive around and perceive objects in the environment. These objects have to transferred from one location to another. To move an object, two robots have to col-

laborate. Therefore, two roles are associated with the task: a helper role and a collector role. The coordination among agents is based on the exchange of tokens, which are associated with the roles of a task. Tokens can be exchanged between agents and whoever owns a token can participate in the task in the associated role.

In earlier work [3], we have described an approach for flexible and decentralized task allocation that deals with the problems mentioned above. This approach consists of a continuous *negotiation* protocol where the situation and the allocation of tasks is continuously *reconsidered* until the task is actually started. The approach presented in this paper builds upon the principles developed in that research. However, here we provide a simplified and uniform model that consistently uses fields for task assignment.

3.2 Field-Based Techniques in MASs

Techniques based on *fields* have proven to be valuable in various applications that involve movement of agents in a metric space [7, 15, 12, 16, 13, 14]. In this approach, objects in the environment produce fields, which are propagated in the environment in a certain range. At each position in the metric space these fields have a certain value, forming *potential fields*. These values are typically proportional to the distance from the source of the field. Agents perceive the fields and combine them in a certain way. The behavior of an agent then simply consists of following the combined potential field downhill, that is, agents follow the direction of the *gradient* of the field. As an example, suppose an agent has to move to a goal while avoiding obstacles in the environment. The obstacles emit repulsive fields, while the goal emits an attractive field. The agent combines the fields by adding the values of the attractive and repulsive field. The combined field can be visualized as a landscape, with the goal representing a valley and the obstacles mountains. This combined field is then followed downhill, which can be viewed as a ball rolling down the slopes of the landscape towards the goal. The field-based approach readily extends to task allocation, where each task emits an attractive field and agents follow the strongest field. Competition among agents can be handled by letting the agents emit repulsive fields.

Field-based techniques have also been applied in systems that are inspired by biological phenomena and physics. For example, birds in a flock all fly in the same direction and avoid collision by keeping a fixed distance between each other. An example application for this idea are guards who have to patrol a museum, keeping a certain distance between each other to cover more ground [15]. Shehory et al. [18] describes a physics oriented model for cooperative goal-satisfaction with simple agents in a large-scale cooperative MAS. Different kinds of gradient field approaches have also been used in the context of RoboCup, a recent example is [4].

[12] discusses a variation on the field-based approach where agents construct a field in their direct neighborhood to achieve routing and deadlock avoidance in a simplified AGV systems. Another variation is described in [16], where the MMASS (Multilayered Multi Agent Situated System) model for multiagent coordination is used. In this model, the environment is represented as a multi-layered graph in which the agents can spread abstract fields. In the standard field-based approach, agents combine perceived fields and are constantly guided by the fields, while in MMASS fields are in principle considered independent from each other and are exploited only to trigger one shot reactions.

Pros versus Cons. The field-based approach is an elegant and effective way to tackle problems in a metric environment. The agents can be kept simple, relying on the environment to achieve the system objective. Since no rigorous protocols are used, robustness and openness (agents can easily be added or removed from the system) are achieved. The approach also benefits from adaptivity: the landscape of fields is readily adjusted to changes in the environment. Also, the fields can be interpreted and combined in different ways by different agents (e.g. a location may be a goal for one agent, while another agent needs to avoid it).

However, the field-based approach also suffers from some limitations. Firstly, the approach is difficult to apply when the situation cannot be represented as a metric space. Secondly, local minima can arise, agents can become stuck at a local minimum, leading to sub-optimal solutions. Another problem is that the approach is not easily to combine with cognitive decision making mechanisms. Finally, a disciplined methodology is lacking to predict the overall evolution of the system. With the field-based approach, the solution emerges from the interaction among the agents. Typically, the development of such systems requires intensive parameter tuning.

Maintaining Fields. An important issue when using the field-based approach is the distribution and maintenance of fields within the environment. A possible way to maintain fields is a storage infrastructure to store the different values of fields in different (digitized) positions in the environment. When dealing with a software environment, this requirement is automatically achieved. For example, the field-based approach has been used successfully in games such as in "Quake 3 Arena" [13] to guide the behavior of non-human characters.

When dealing with a physical environment, a distributed storage capability could be physically supported by the environment. For example in [15], the field-based approach is applied in guiding tourists in a museum. The tourists are provided with a software agent running on some wireless handheld device, giving suggestions on how and where to move. A computer network with a topology that mimics the topology of the museum plan is embedded in the museum walls. The hosts in the network are associated to each room and are capable of communicating with each other and the mobile devices in their proximity. Fields can be injected at a host, after which they will be diffused hop-by-hop across the network, modulating the values of the fields as necessary. The latter is achieved by supporting middleware, such a TOTA (Tuples On The Air) [14].

If no environmental storage infrastructure is available, fields can still be realized by broadcasting information messages, which contain the position of the source of the field and its initial strength. Agents receive these messages and calculate the value of the field on their position. In this paper we have applied this approach since a typical AGV system provides no distributed storage infrastructure external to the mobile AGV machines.

4. FIELD-BASED TASK ASSIGNMENT

In this section we explain the approach for field-based task assignment in AGV transportation systems. Similar to the approach in [3], task assignment is *delayed* until the AGV actually reaches the pick location of the transport, allowing the system to react flexibly and adaptively to new transports that enter the system. We start with a high-level explanation of the approach, followed by an overview of the software architecture.

4.1 High-Level Overview

The basic idea of field-based task assignment is to let each idle AGVs follow the gradient of a field that guides it toward a load that has to be transported. There are two types of fields in the system: (1) transports emit fields into the environment that attract idle AGVs; (2) to avoid multiple AGVs driving towards the same transport, AGVs emit repulsive fields. AGVs combine received fields and follow the gradient of the combined fields, that guide them towards pick locations of transports. The AGVs continuously reconsider the situation of the environment and task assignment is delayed

until the load is picked, which benefits the flexibility of the system. To explain the field-based approach for taks assignment, we use the scenario depicted in Fig. 1.

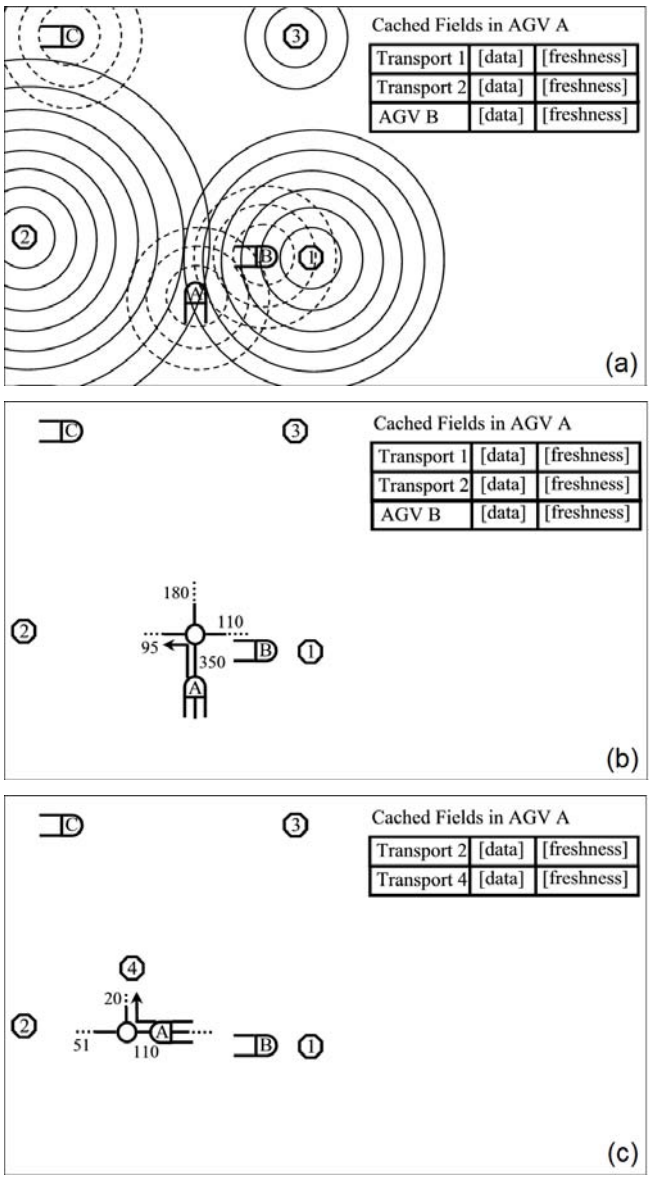

Figure 1: ***Example scenario of the developed approach. The full circles represent transport fields, the dotted circles AGV fields. For clarity, we have not drawn the fields in (b) and (c).***

- **AGV agents and transport agents.** Task assignment is achieved by the interaction between two kinds of agents: *AGV agents* and *transport agents*. The AGV agents correspond to the physical vehicles and the transport agents represent transports. Whenever a new transport enters the system, a corresponding transport agent is created. The lifetime of the transport agents ends when the associated load is delivered. *Conceptually* a transport agent resides at the pick location of the associated transport. On which physical computer system(s) the transport agents reside is not important for the discussion in this paper, the only important issue is that the agents operate independently and therefore can be distributed across several computer systems.

- **Agents emit fields.** Both AGV and transport agents emit fields, called *AGV fields* and *transport fields* respectively, see Fig. 1-a. Fields have a certain range and contain information about the source agent. AGV fields have a fixed range, while the range of transport fields is variable. Fields are refreshed at regular times, according to a predefined refresh rate.

- **AGV agents store received fields.** When an AGV agent perceives fields, it stores the data contained in these fields in a *field-cache*. The field-case consists of a number of cache-entries. Each cache entry contains the source of the received field (an ID), the most recent data contained in that field and a *freshness*. The freshness is a measure of how up-to-date the cached data is. Fig. 1 shows the field-cache of AGV A through the scenario.

- **AGV agents construct calculated-fields to decide their movement.** An AGV agent constructs a *calculated-field* to decide in which direction to drive from a station. A calculated-field is a combination of the received fields, which are stored in the field-cache. The lower the freshness of a cache-entry, the lower the influence of the associated field on the calculated-field.

  The calculated-field is constructed locally and temporarily from the last selected target station (this is the last station that is added to the look ahead buffer, we further clarify this below) and contains values for each outgoing segment. An AGV agent follows the calculated-field in the direction of the smallest value. This can be considered as following the calculated-field downhill, in the direction of the *gradient*.

  Transport fields have an *attractive* influence on the calculated-field, which results in AGVs driving towards unassigned transports. However, it is undesirable that many AGVs drive to the same transport, since they will obstruct each other and travel superfluous distance. To remedy this, AGV fields have a *repulsive* influence.

  In Fig. 1-b, AGV A constructs a calculated-field on a station. Although transport 1 is closer, the calculated-field will guide AGV A towards transport 2. This is the result of the repulsive effect of AGV B. Notice that it would have been ineffective for AGV A to drive towards transport 1, since AGV B is closer and likely will reach the transport first.

- **Task assignment occurs at pick up.** Task assignment is delayed until an AGV actually reaches the pick location and picks up the load. This results in a greater flexibility with respect to task assignment, in comparison for example to the CNET protocol, which assigns a transport as soon as it enters the system. By delaying task assignment, the field-based approach can cope with unforseen situation like transports which suddenly popup, as illustrated in Fig. 1-c. While AGV A is driving towards transport 2, a new transport (transport 4) appears close to AGV A. Since no transport has been assigned to AGV A yet, it can drive towards the closer transport 4.

4.2 Software Architecture

The software architecture of the AGV system is shown in Fig. 2. This architecture is based on a reference model for situated MAS that is described in [20, 22, 23]. Successively, we explain the different agent components. The architecture of the environment is not further discussed here.

- **Field-cache**: This module of the AGV agent stores the information of fields received from other AGV agents and transport agents in cache-entries. A freshness is associated with each cache-entry, which is a measurement of how up-to-date the entry is.

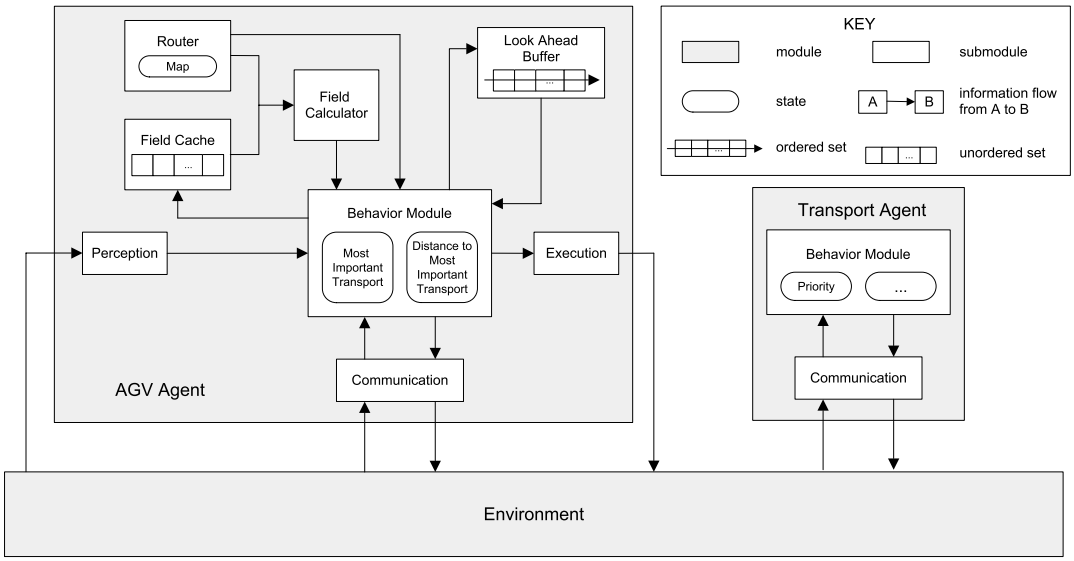

Figure 2: ***The architecture of the AGV transportation system.***

- **Router**: This module contains a global map of the stations and segments. The router is used by the AGV agent to calculate paths and distances from one station to another. We used a *static* router that uses the A* algorithm [8]. However, the approach discussed in this paper is fully compatible with a *dynamic* router that would take dynamic runtime information into account such as traffic distribution or blocked segments.

- **Look-ahead buffer**: As explained in section 2, an AGV has to make routing decisions over a certain distance in advance to allow save and smooth movement. This distance, called the *look-ahead*, is not a straight line distance, but is measured discretely, in terms of stations and segments of the environment map. The look-ahead buffer of an AGV agent is an ordered array that holds these routing decisions. It represents a path that the AGV will follow with certainty, unless something catastrophical (such as a crash) happens.

- **Field calculator**: This module constructs a calculated-field locally and temporarily from the last selected target station (i.e. the last station added to the look-ahead buffer) by combining the received fields, which are stored in the field-cache. The higher the freshness of a cache-entry, the more influence the field associated with the cache-entry will have on the construction of the calculated-field. Thus, although still used, less importance is given to outdated information. The field calculator makes use of the router to calculate the values of the calculated-field on different positions. The gradient of the calculated-field is used as driving direction on the target station.

- **Behaviour module**: The behavior module continuously reconsiders the dynamic conditions in the environment and selects appropriate actions to achieve the agent's goals. The behaviour module of an AGV agent also maintains the *current most important transport* and the *distance to the most important transport*. These values are used to achieve the repulsive influence of AGV agents, as described above. The behaviour module of a transport agent maintains, among other data, the current *priority* of the transport and the *field range*. To avoid starvation of the transport, the priority grows over time. The field-range of the transport agent is a function of time and the number of interested AGV agents. The following two high-level descriptions summarize the behavior of the agents during task assignment:

```
{Behavior procedure of the AGV agent}
while idle
 do repeat with constant frequency {
  1. Do communication
   1.1 Send out field
   1.2 Accept received fields and enter
       them in the field-cache
  2. Fill the look-ahead buffer (if needed)
  3. Perform an action
}

{Behavior procedure of the transport agent}
while not assigned
 do repeat with constant frequency {
  1. Do communication
   1.1 Send out field
   1.2 Accept received messages
  2. Calculate priority and field-range
}
```

5. TEST RESULTS

This section discusses the main test results obtained from applying the field-based task assignment approach. For a detailed discussion of the AGV simulator used in the tests we refer to [1, 17]. We start this section with explaining the test setting and subsequently we elaborate on the test results.

5.1 Setting

Map. All tests were performed on the map of a real layout that has been implemented by Egemin at EuroBaltic, see Fig. 3. The size of the physical layout is 134 m x 134 m. The map has 56 pick and 50 drop locations. The six locations at the lower side of the map are only pick locations, all other locations are pick and drop locations.

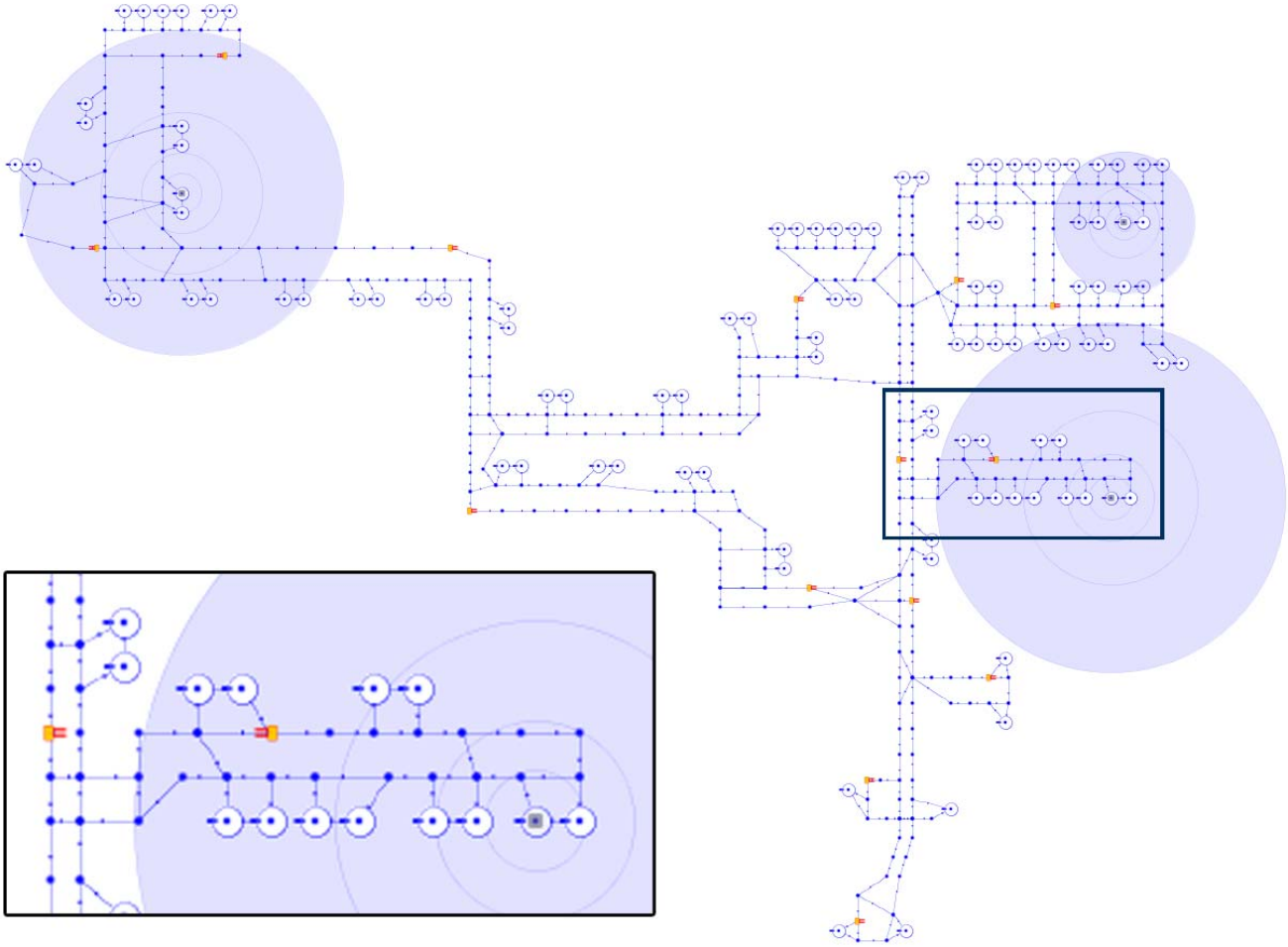

Figure 3: ***Map used in the tests.***

To allow running tests with the current implementation of the AGV simulator, we made a number of small modifications to the map, e.g., all curved segments were changed to straight lines.

Transport profile and AGVs. We used a standard transport test profile that is been used by Egemin for testing purposes. This profile generates 140 transports with a random pick location and a random drop location per simulation run, corresponding to the number of transports that are generated in 1 hour real time.

In the simulation, we used 14 AGVs just as in the real application. The average speed of driving AGVs is 0.7 m/s, while pick and drop actions take an average amount of time of 5 s.

Infrastructure. The AGV simulator and the multiagent system used for testing the field-based approach for task assignment are written in Java. An explanation of the working of the AGV simulator, the source code, as well as the tests results are available on the web [1]. The simulator uses a framework for time management [9] to make sure the simulation results are independent of performance characteristics of the execution platform, working load of the processor, or amount of memory.

To allow running the huge amount of simulations, we used a cluster of 40 machines: P4 2Ghz, 512MB RAM, Debian Stable 3.0. Testing one set of parameter values took approximately 30 hours of simulation time.

Metrics. Every simulation was run for 50000 timesteps, corresponding to approximately 1 hour real time, i.e. one time step represents 20 ms in real time. All test results displayed in the paper are average values over 20 simulation runs. Performance is measured in terms of throughput and reaction time. Additionally, the amount of communication needed is measured.

Reference algorithm: CNET. We used CNET as a reference algorithm to compare the field-based approach. With CNET each transport that enters the system is assigned as soon as possible to the most suitable AGV (i.e., an idle AGV for which the cost to reach the pick location is minimal). When transports can not be assigned immediately, they enter a waiting status. All waiting transports are ordered by priority, and this priority determines the order in which transports are assigned. The priority of transports grows over time, the same way as in the field-based approach.

5.2 Results

A number of tests have been performed to measure the performance of the field-based approach, both for tuning the parameters of the field-based approach and for comparison with CNET under varying circumstances and parameters. Due to space limitation we limit the discussion to the main test results. For a detailed discussion of the remaining test results we refer to [17].

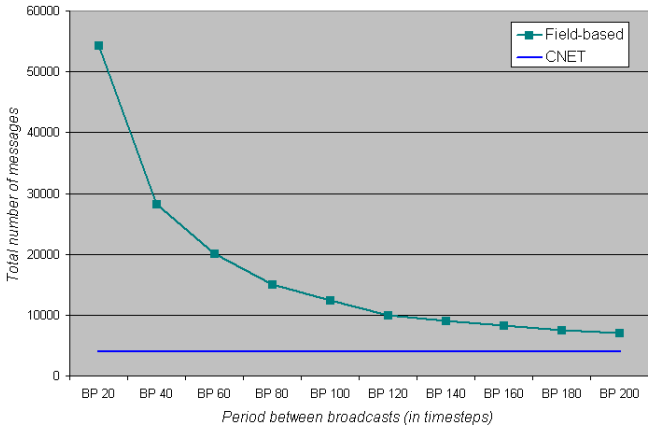

Figure 4: ***Amount of messages being sent.***

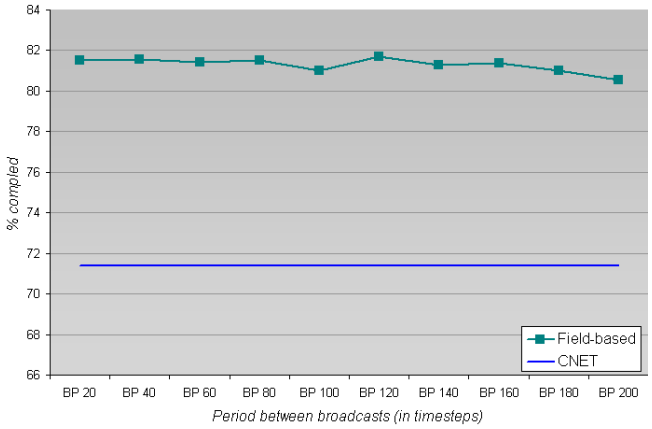

Figure 5: ***Percentage of completed transports.***

5.2.1 *Refreshment of Fields*

As stated before, maintenance of fields in our approach is achieved by periodically broadcasting the status of the fields (both position and strength). Obviously, the period between subsequent broadcasts strongly affects the amount of messages being sent. Making the period too short will produce a huge number of useless messages, but each agent in the system will have up-to-date information. On the other hand, if the broadcast period is too long, AGVs may have outdated information about the fields and probably miss some opportunities.

Fig. 4 shows the expected decrease in number of messages sent if the period between two broadcasts increases. BR20 corresponds to a field refresh each 20 time steps, i.e. each agent broadcasts its field status every second. With a field refresh each 200 time steps (BP200, i.e. every 10 s) the number of messages sent with the field-based approach is 1.7 times higher then with CNET. Of course what is interesting are the implications of reducing the field refresh on the performance of the system.

5.2.2 *Performance and Field Refresh*

Fig. 5 depicts the percentage of transports handled as a function of the field refresh rate. It can be seen that the percentage of completed tasks fluctuates around 81,5% and slowly decreases, the difference between BR20 and BR200 is only 1% but still significantly

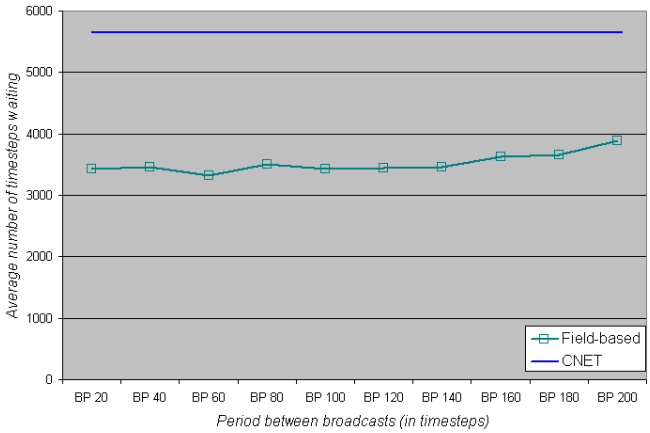

Figure 6: ***Average wait time for transports.***

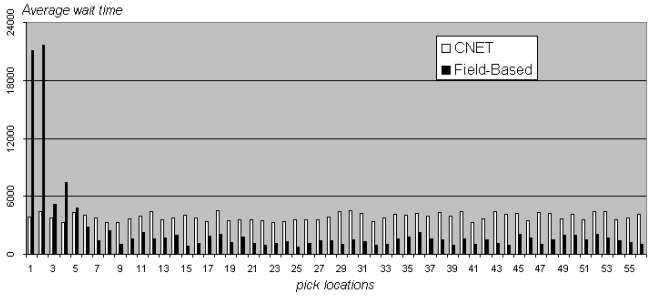

Figure 7: ***Average wait time for transports per pick location.***

better then the CNET approach. Fig. 6 illustrates that the average waiting time slowly increases with lower refresh rates, here the difference between BR20 and BR200 is 14%, which is still 31% better then the CNET approach.

These results clearly illustrate that communication overhead can be reduced by using a longer broadcast period, without significant performance loss. Overall, the throughput of the field-based approach is 10% higher than the throughput of CNET. The average waiting time of a transport is 39% lower, while the average number of transports waiting at each time step was around 20% lower for the field-based approach. [17] elaborates on the distance travelled by the AGVs. On average, the travelled distance of all AGVs with the field-based approach was 33% lower compared to CNET which is a result of a more optimal allocation of tasks.

5.2.3 *Average Waiting Time per Pick Location*

Although the average waiting time for transports is significantly better for the field-based approach compared to the reference algorithm, it is interesting to compare the average waiting time for transports per pick location.

Fig. 7 shows the average wait time for transports grouped by pick location. Clearly, the CNET reference algorithm achieves a more equal distribution. In particular, the wait times for pick locations 1 to 5 are significantly higher for the field-based approach. This drawback can be explained as follows: because the pick locations 1 to 5 are far away from the main traffic in the warehouse, the chance an AGV will be close to the pick location is significantly lower and this decreases the chance for immediately attraction an idle AGV when a new transport pops up. Starvation is prevented since the priority

of the transports on the remote locations gradually increasing when the load is not picked. It simply takes a longer time for the field to "grow" and attract AGVs compared to the immediate assignment of an AGV in the CNET protocol. A possible remedy to this problem is to increase the strength of fields of transports on isolated locations right from the moment the transport is created.

Finally, gradient field-based approaches are known to suffer from local minima. To deal with this problem agents do not use straight distances to sources of fields to construct calculated fields, but use a router that takes into account real distances along the paths on the map. As a result, we experienced little problems with local minima.

6. CONCLUSION

In this paper we presented a field-based approach for task assignment in AGV transportation systems. In this approach, each idle AGV is guided toward a load of an unassigned transport by following the gradient of a field that combines attracting fields received from transports and repulsing fields received from competing AGVs. By delaying the definitive assignment of a transport until the load is finally picked the approach achieves the required flexibility to exploit opportunities that may arise while AGVs search for transports in the highly dynamic environment.

Extensive tests show that the field-based approach outperforms standard CNET for throughput as well as for average waiting time of transports. On the other hand, the approach requires a limited amount of additional bandwidth, and the waiting time for isolated pick locations is higher compared to CNET. We experienced little problems with the traditional difficulties of field-based approaches, including parameter tuning and problems with local minima. As an overall conclusion, field-based task assignment is a promising approach in the domain of AGV transportation systems.

As future work, we aim to compare the field-based approach with other flexible approaches for task assignment. Currently we study a dynamic version of CNET in which transport agents and idle AGV agents continuously reconsider the assignment of transports until loads are picked, aiming to exploit opportunities that arise in the environment. Another interesting venue for future work, is to separate the management of fields from the agent logic by using a *virtual environment* as proposed in [21]. In this approach, agents may simply perceive fields in the virtual environment, while this latter–supported by middleware–takes the burden of managing the spreading of fields. This model improves separation of concerns by making the environment responsible for managing fields, which responsibility conceptually belongs to the environment.

7. REFERENCES

[1] Automatic Guided Vehicle Simulator, K.U.Leuven, 2005. http://www.cs.kuleuven.ac.be/∼distrinet/taskforces /agentwise/agvsimulator/.

[2] S. Aknine, S. Pinson, and M. F. Shakun. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8:5–45, 2004.

[3] N. Boucke, D. Weyns, T. Holvoet, and K. Mertens. Decentralized allocation of tasks with delayed commencement. In *Proceedings of European Workshop on Multiagent Systems, Barcelone, Spain*, 2004.

[4] G. Buchman, D. Cohen, P. Vernaza, and D. Lee. The University of Pennsylvania Robocup 2005 Legged Soccer Team. *http://www.cis.upenn.edu/robocup/UPenn05.pdf*, 2005.

[5] Egemin Modular Controls Concept. EMC ^{2} project, IWT, Belgium. *http://emc2.egemin.com/*, 2005.

[6] A. Farinelli, L. Iocchi, D. Nardi, and V. Ziparo. Task Assignment with Dynamic Perception and Constrained Tasks in a Multi-Robot System. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[7] J. Ferber. *Multiagent systems: An introduction to distributed artificial intelligence*. Addison-Wesley, London, UK, 1999.

[8] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 1968.

[9] A. Helleboogh, T. Holvoet, D. Weyns, and Y. Berbers. Extending time management support for multi-agent systems. In *Multiagent and Multiagent-based Simulation, New York, USA*, 2005.

[10] T. Knabe, M. Schillo, and K. Fischer. Improvements to the FIPA Contract Net Protocol for Performance Increase and Cascading Applications. *Proceedings of the Workshop on Multiagent Interoperability*, 2002.

[11] T. Knabe, M. Schillo, and K. Fischer. Interorganizational Networks as Patterns for Selforganizing Multiagent Systems. *Autonomous Agents and Multiagent Systems*, 2003.

[12] L.Breton, S. Maza, and P. Castagna. Simulation multi-agent de systèmes d'AGVs: comparaison avec une approche prédictive. 5^e *Conférence Francophone de Modélisation et Simulation*, 2004.

[13] M. Mamei and F. Zambonelli. Motion coordination in the quake 3 arena environment: a field-based approach. *First International Workshop on Environments for Multiagent Systems*, 2003.

[14] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *Proceedings of the 2nd International Conference on Pervasive Computing and Communications*. IEEE Computer Society, Washington, DC, USA, 2004.

[15] M. Mamei, F. Zambonelli, and L. Leonardi. Co-Fields: A Physicall Inspired Approach to Distributed Motion Coordination. *IEEE Pervasive Computing*, 2004.

[16] F. D. Paoli and G. Vizzari. Context dependent management of field diffusion: an experimental framework. *Workshop Dagli Oggetti agli Agenti, Villasimius, Italy*, 2002.

[17] W. Schols, T. Holvoet, N. Boucke, and D. Weyns. Gradient Field Based Transport Assignment in AGV Systems. In *CW-425, Technical Report, Departement of Computer Science, Katholieke Universiteit Leuven, Belgium. http://www.cs.kuleuven.ac.be/publicaties/rapporten/CW/2005/.*

[18] O. Shehory, S. Kraus, and O. Yadgar. Emergent cooperative goal-satisfaction in large scale automated-agent systems. *Artificial Intelligence*, 110(1):1–55, 1999.

[19] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers, C-29(12)*, 1980.

[20] D. Weyns and T. Holvoet. Formal Model for Situated Multiagent Systems. *Fundamenta Informaticae*, 63(2-3), 2004.

[21] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In *4th Joint Conference on Autonomous Agents and Multiagent Systems, Industry Track*, 2005.

[22] D. Weyns, E. Steegmans, and T. Holvoet. Protocol based communication for situated multiagent systems. In *3th Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

[23] D. Weyns, E. Steegmans, and T. Holvoet. Towards active perception for situated multi-agent systems. *Applied Artificial Intelligence*, 18(8-9), 2004.