Specifying Desired Behaviour in Hybrid Central/Self-Organising Multi-Agent Systems

Yaser Chaaban and Christian Müller-Schloer
Institute of Systems Engineering
Leibniz University of Hanover
Hanover, Germany
chaaban,cms@sra.uni-hannover.de

***Abstract*—Path planning for multiple agents in a common environment is a challenging research problem. In previous papers, we introduced a hybrid architecture solving the conflict between a central unit (an observer and a controller) and decentralised autonomous agents. This architecture should have mechanisms for planning, but only as recommendation, and control decisions. In this paper, we focus on planning of the desired behaviour of agents in a shared environment. The scenario used in this work is a traffic intersection without traffic lights. Therefore, an A*-algorithm was adapted for the path planning. Consequently, the designed algorithm calculates collision-free trajectories (central planning) for all agents (vehicles) in a shared environment (the centre of the intersection) enabling them to avoid collisions. The experimental results demonstrated a high performance of our adapted A*-algorithm.**

Keywords- Path planning; A*-algorithm; Multi-Agent Systems; autonomous vehicles; Hybrid Coordination; Organic Computing

I. INTRODUCTION

Organic systems [1] or autonomic systems try to realise quality in several aspects of system engineering [2][3]. The wide range of properties of organic systems can be used to establish the vital concept of "controlled self-organisation". Also, organic systems use the "controlled self-organisation" design paradigm, in which the unwanted behaviour should be prevented, whereas the desired behaviour should be rewarded.

In prior papers [4][5][6][7], we introduced the interdisciplinary methodology, “Robust Multi-Agent System” (RobustMAS). RobustMAS uses a central component that performs the desired behaviour (trajectories), where this planned behaviour is recommended to agents. RobustMAS applies a hybrid solution (central/self-organising), where collision-free trajectories for agents (the desired behaviour) are calculated by a central component and then given to agents only as a recommendation. Here, despite the autonomous behaviour of agents, they always get the best possible (desired) trajectories from the central unit.

In this context, a traffic intersection without traffic lights was chosen as a main testbed to apply the hybrid approach (RobustMAS), where autonomous agents are autonomous vehicles, and the controller of the intersection is the central unit. However, the basic idea of a hybrid approach is applicable for other systems as well.

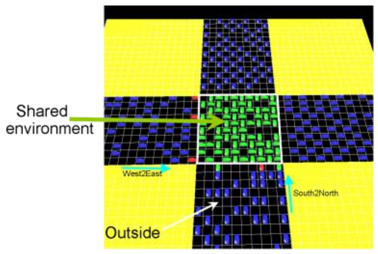

Figure 1. The traffic intersection without traffic lights

Figure 1 shows a screenshot from our project. In this regard, we presented the desired system architecture in [4][5]. This architecture was an observer/controller (O/C) architecture adapted to the scenario of traffic intersection.

In this paper, we concentrate on planning of the desired behaviour (trajectories) of agents (vehicles) in a shared environment (traffic intersection).

This paper is organised as follows. Section 2 presents a survey of related work concerning path planning. In Section 3 the realisation of the approach is discussed. This realisation includes: path planning, A*-algorithm, trajectories, an adapted A*-algorithm and virtual obstacles respectively. Section 4 introduces the evaluation of the system performance by means of experimental results. Section 5 draws the conclusion of this work. Finally, the future work is explicated in Section 6.

II. STATE OF THE ART

In the literature, there are many works concerning path planning, where common path search algorithms are investigated. The problem of path planning for multiple agents (robots) has been discussed in various papers in order to coordinate the movements of the agents [8][9][10]. There are various approaches to solve this problem. Two well-known approaches are: the coordination technique and an A*-based path planning technique [9]. The coordination technique [11] arranges and discovers the optimal paths of the individual agents (robots) and then computes a schedule how the robots have to traverse these trajectories. The A*-based technique applies the A* search algorithm (a graph search algorithm that finds the least-cost path from a given initial node to one goal node) to work out independent planning of the paths for the individual robots in their configuration time-spaces, which extends the configuration space of the robot by a time axis. In [9], a series of experiments have been performed to compare these approaches. These experiments demonstrate that the A*-based technique significantly outperforms the coordination technique. In many cases, the authors were convinced that the A*-based technique is much more efficient because of the independent planning of the paths for the individual robots in the time-spaces configuration. These experiments show also that the A*-based approach is well suited to control the motions of a team of robots in various environments and illustrate its advantages over the coordination technique.

III. The Approach

In our approach, we model vehicles as agents. Also, we define the term “shared environment” as the centre of the intersection.

An intersection manager is responsible for coordinating tasks. It performs first a path planning to determine collision-free trajectories for the vehicles (central). This path planning is given to vehicles as a recommendation. Vehicles are modelled as agents.

For the path planning, common path search algorithms are investigated. Particularly interesting here is the A*-algorithm. The path planning is considered as a resource allocation problem (Resource Allocation Conflict), where several agents move in a shared environment and have to avoid collisions. The implementation should be carried out under consideration of virtual obstacles. Virtual obstacles model blocked surfaces, restricted areas (prohibited allocations of resources), which may arise as a result of reservations, accidents or other obstructions.

In this paper, we assume that all vehicles obey their planned trajectories (plan) and thus no deviations from the plan will occur. In addition, there are no accidents in the intersection. This means that everything is as planned.

A. *Path planning*

This section presents the realisation and requirement of path planning and illustrates the resulting trajectories. Accordingly, the adapted A*-algorithm to calculate collision-free trajectories for all agents is introduced using virtual obstacles.

When every agent has its unique path from one point to another, no conflict is possible when no unexpected errors or disturbances occur during movement of agents. In order to plan such unique paths for multiple agents that move in a shared space, global knowledge and centralised control will be needed so that it will be easy to prevent conflicts.

Path planning in this work is the applied coordination mechanism to solve the problem of resource sharing wherever multiple agents cross the shared environment avoiding collisions. Path planning delivers collision-free trajectories for all participants in this multi-agent system. The behaviour of an agent outside the shared environment do not need path planning, because an agent outside the shared environment has only local rules, through which it tries to move forward avoiding collisions with other agents. Path planning has to be done only for agents inside the shared environment.

When an agent arrives at a border of the shared environment (Figure 1), it sends a message (request) to the controller (intersection). The path planning unit of the controller has to reply to this message thereby the agent can cross the shared environment safely provided no unexpected errors or deviations from the plan occur within this process.

When the path planning unit of the controller receives a request from an agent (vehicle), it simulates the trip of that agent through the shared environment taking into consideration the presence of other agents and the geometry of the shared environment (intersection) in the configuration time-spaces. It calculates an appropriate trajectory and sends it to the agent. Furthermore, the calculated trajectory is stored in the trajectory memory.

The enquiring agent gets its trajectory, which guarantees a coordinated behaviour with the other agents in order to avoid traffic jams in the intersection.

B. *A*-algorithm*

Since our work uses the A*-based technique, which employs the A*-algorithm, this section presents the A* procedure described by Nilsson et al. in [12]. As mentioned above, A*-algorithm is a search algorithm to obtain the optimal path (minimum-cost path according to a given cost function) from a given start state to a target state in a graph. In order to build only paths that lead towards the target state, A* uses priorities assigned to each path. The priority of a path n is determined by the cost function: $f(n)=g(n)+h(n)$. It should be mentioned that using a priority queue to store paths through the graph (already visited nodes) together with their related A* costs is the most common implementation of the A*-algorithm. For this purpose, the lower the A* cost, i.e., the $f(n)$ cost, of the node n, the higher the priority assigned to this node.

Here, $f(n)$ is the total A* cost of the path from the start state (start node) until the current state (current node) n, where $f(n)$ is composed of $g(n)$ and $h(n)$. First, $g(n)$ represents the accumulated costs of reaching the state n from the start state. Second, $h(n)$ is the estimated cost of reaching the goal state from the state n. The estimated cost is called heuristics. The cost function $f(n)$ plays a main role in finding optimal paths, because A* takes into account the distance already travelled, the $g(n)$ function. Therefore, A* will only obtain the shortest path, if it exists, when a good heuristics is selected. The algorithm 1 gives an overview of how the A*-algorithm works.

C. *Trajectories*

A trajectory in our work represents the path of an agent only inside the shared environment (inside the intersection). The controller plans trajectories for all agents in the system, which have to be collision-free. If all agents comply with their planned trajectories, then the throughput of the system would be better (the intersection will be covered optimally by the vehicles), because RobustMAS uses a central algorithm in order to plan the trajectories. Here, the central planning algorithm (A*-algorithm) has a global view of all available resources (cells in the intersection) that can be allocated to the agents.

The agents get their planned trajectories only as recommendation from the controller, because they can behave in a fully autonomous way.

The memory of all trajectories serves the observer to detect any deviations from the planned trajectories occurred in the system, where the observer compares the actual travelled trajectories to this memory.

A trajectory is modelled as a vector (n *-tupel*) of space-time points, where each point has its coordinates (x_i, y_i) that can be reserved at time t_i :

$$trajectory = \{(x_1, y_1, t_1), \ldots\ldots, (x_n, y_n, t_n)\} \text{ with } 1 \le i \le n;\ i,n \in \mathbb{N} \quad (1)$$

D. An adapted A*-algorithm

In this paper, the A*-procedure for path planning of agents is applied and the minimum-cost path in its three-dimensional configuration time-space is searched. However this A*-based procedure has been adapted for the requirements of the used application scenario “intersection without traffic lights”, because a vehicle can only take a “rational” path, whereas an agent (e.g., robot) can take any calculated path. Here, the term "rational" denotes the fact that a vehicle carries out a goal-directed motion along a rational (most reasonable) path in the intersection when it moves towards its target. This path will be a straight or concave trajectory (or sections of a trajectory) with respect to the travel direction as depicted in Figure 2.

However, robots can follow an arbitrary (winding) path. As a result, due to the use of the A*-algorithm not adapted to a traffic intersection, “non-rational” path (or sections of the path) from one waypoint to the target can be built and consequently used, where vehicles can not take such paths in the centre of an intersection to reach their targets. Examples of “irrational” paths are due to repeated zigzag movements, or back and forth movements (crisscross).

Figure 3 shows how A* is used for the problem supported by an example. Here, the trajectory of the vehicle consists of six points. Every point has its (x, y, t) in the three-dimensional configuration time-space as follows:

$$\begin{aligned} trajectory = \{&(x_1, y_1, t_1), (x_2, y_2, t_2), (x_3, y_3, t_3), (x_4, y_4, t_4), \\ &(x_5, y_5, t_5), (x_6, y_6, t_6)\} \end{aligned} \quad (2)$$

Compared to the A*-algorithm described above, the adapted A*-algorithm, which is used in this work, has the following features:

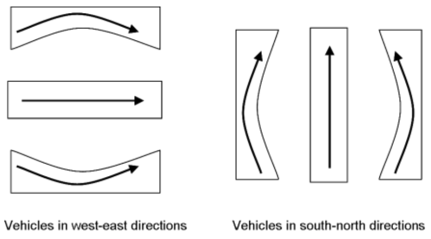
Vehicles in west-east directions
Vehicles in south-north directions

Figure 2. Rational paths of vehicles with respect to the travel direction

Algorithm 1: Overview of the A*-algorithm

```
A-Star (startNode, goalNode)
BEGIN
Initialise (G, Q, startNode); //G:graph, Q:priority queue
CostUntilNow [startNode] = 0;
optimalPathUntilNow [startNode] = startNode;
// Insert the start node in Q with the initial cost f=0
Q = addToQueue (startNode), f (startNode) = 0;
while not isEmpty (Q) do
    bestNode = returnFirstElementOfQueue (Q);
    if (bestNode = = goalNode) then
        // optimal path is given by optimalPathUntilNow []
        return optimalPathUntilNow [];
    end if
    // for each neighbour n of bestNode
    for all n in successors (bestNode) do
        // IS the path to n is shorter than the current way
        if (CostUntilNow[n] > CostUntilNow [bestNode] +
        CostDistBetweenNeighbour (bestNode, n)) then
        // Update the costs given by CostDistBetweenNeighbour
        CostUntilNow [n] = CostUntilNow [bestNode] +
        CostDistBetweenNeighbour (bestNode, n);
        optimalPathUntilNow [n] = bestNode;
            if (n in Q) then
                update f(n)= CostUntilNow[n] + h(n,
            goalNode) in Q;
            else
                Q = addToQueue (n), f(n) = CostUntilNow [n]
                + h (n, goalNode);
            end if
        end if
    end for
end while
// No path could be found.
return failure;
END
```

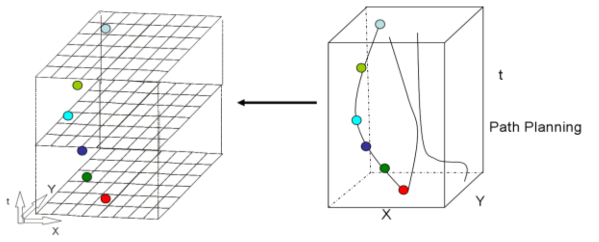

Figure 3. An adapted A*-algorithm used for the problem of path planning in the three- dimensional configuration time-space.

- The function **BuildVirtualObstacles (n)**: It uses this function in order to build virtual obstacles into the path from the current node n to the goal node, because a vehicle can only follow a “rational” path as explained above. Virtual obstacles are blocked areas, which can not be crossed by vehicles. For details see section E (Virtual obstacles).
- It plans independent paths for the individual vehicles in their three-dimensional configuration time-spaces. Thus, reservation of space-time points (x_i, y_i, t_i) is the key step of the adapted A*, where each node (space-time point, tile or cell in the intersection) of the graph that has its coordinates (x_i, y_i) can be reserved by one agent A_i at time t_i . For this purpose, the adapted A*-algorithm uses the function **isCellReserved (n, time)**. This function tests whether the node $n(x, y)$ has been already reserved for another agent for a specific time, where the parameter “time” represents the time at which the agent, for which A* is looking for the best trajectory,

will reach the node $n(x,y)$ according to its planned trajectory so far. So, RobustMAS considers the problem of path planning for teams of agents.

- It provides the possibility to react to potential deviations of the agents from their planned trajectories during the plan execution. Deviations from the planned trajectories are detected by the observer of the O/C architecture, where the controller is informed of it. Consequently, the adapted A* re-plans the affected trajectories using the function **replanNewTrajectoriesOfAffectedAgents()**. Moreover, it takes into account the presence of disturbances (i.e., accidents in the intersections) by computing the paths.
- The heuristics used in the adapted algorithm for the estimated cost of reaching the goal state is based on the straight-line Euclidean distance from any given state (a node in the graph) to the goal state:

$$\min\|(x_s,y_s)-(x_g,y_g)\|;(x_s,y_s){:}\textit{start state},\ (x_g,y_g){:}\textit{goal state} \quad (3)$$

  This heuristics (a heuristic estimation of the distance in the case of path planning) will enable definitely A* finding the shortest path, if it exists, where the search will be limited to selected collections of the state space. Thus, a heuristic estimate of the distance to be travelled may be the straight-line distance between two states in a shared environment, so that optimal paths can be planned.

E. *Virtual obstacles*

The implementation of the adapted A*-algorithm in RobustMAS has been carried out under consideration of virtual obstacles. Virtual obstacles have been adopted, where blocked surfaces should not be considered by the planner. Virtual obstacles model blocked surfaces, restricted areas, which may arise as a result of reservations, accidents or other obstructions. In addition, virtual obstacles can be used for traffic control. Figure 4 shows the shape of the blocking surfaces (virtual obstacles), which are used by RobustMAS in order to plan trajectories using an adapted A*-algorithm.

IV. PERFORMANCE EVALUATION

In this section, we present an initial evaluation of our algorithm using the model of a traffic intersection, which was designed and described in our earlier papers [4][5].

Since the path planning algorithm plays an important role in our traffic system to achieve high performance, an evaluation of this algorithm is required under different test scenarios considering various loads of vehicles, where no deviations from plan and no accidents occur in the system. In our earlier papers, handling of deviations from planned (desired) behaviour was studied in [6], whereas handling of disturbances (accidents) was considered in [7].

The evaluation of the concept was carried out based on the basic metrics: throughput and mean waiting time. Throughput here is the total number of vehicles that left the intersection (simulation area) over time, whereas the mean waiting time is the average time (ticks or iterations) needed by vehicles to traverse the intersection:

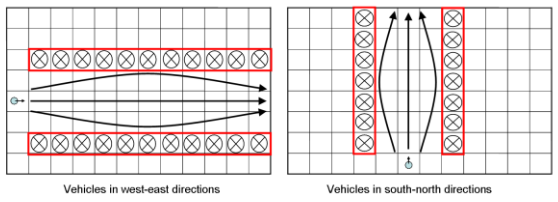
Vehicles in west-east directions
Vehicles in south-north directions

Figure 4. Blocking surfaces (virtual obstacles)

$$MWT_i = \frac{1}{v}\sum_{k=1}^{v} W_{k,i} \quad (4)$$

Where MWT_i is the mean waiting time of the system at the time (tick) i, $W_{k,i}$ is the waiting time of the vehicle k at the time i, and v is the total amount of vehicles.

A. *Evaluation scenarios*

Four different evaluation scenarios are used to measure and compare the system performance, which results from change of values of the following two simulation parameters: $V_{\max}$ is the maximum number of vehicles in each direction, and TL is the production rate of vehicles in each direction (traffic level or traffic flow rate). The four different evaluation scenarios ensure that the system performance in various combinations of the parameter remains effective even when the intersection is very busy, especially during rush hour (during morning and afternoon peak traffic).

The used metrics have been measured in an interval between 0 und 3000 ticks (time steps). As shown in Figure 1 two traffic flows with orthogonal directions are taken into account: $W2E$ (West2East) and $S2N$ (South2North).

Table I shows the resulting four evaluation scenarios. Here, "equal TL" means that the traffic flow rates of vehicles in each direction are the same, while these rates in each direction are different in the case of "not equal TL". Similarly, the "equal $V_{\max}$" and "not equal $V_{\max}$" can be expressed by the parameter "maximum number of vehicles". The values of both simulation parameters were chosen in such a way that a wide spectrum of traffic volumes can be covered (low, medium, high and extreme traffic volumes).

For example, in evaluation scenario I (Equal $V_{\max}$ – Equal TL), the throughput and the mean waiting time of the system have been measured in the case that the traffic flow rates (traffic levels) of vehicles in south-north and west-east directions is equal, namely 5 vehicles/tick, where the measurement has been repeated in the cases that the maximum number of vehicles in each direction is equal, namely 20, 40, 80, 100, and 500 vehicles. The case of 500 vehicles in every direction is an extreme case, where the maximum number of vehicles is greater than the capacity of the intersection (very busy intersection). Here, the intersection is a 10x10 grid of reservation tiles. Similarly, other evaluation scenarios II, III, and IV can be expressed.

B. *Results of throughput measurement*

Figure 5 shows the system throughput (#Vehicles/tick) for each evaluation scenario that was measured in an interval between 0 und 3000 ticks.

TABLE I. THE FOUR EVALUATION SCENARIOS (TWO TRAFFIC FLOWS WITH ORTHOGONAL DIRECTIONS)

Scenarios	#Max. Number of Vehicles (V_{max})	# Traffic levels (TL) (Traffic flow rates)
I (Equal-Equal)	N2S = W2E	N2S = W2E
II (Equal-Not Equal)	N2S = W2E	N2S ≠ W2E
III (Not Equal-Equal)	N2S ≠ W2E	N2S = W2E
IV (Not Equal-Not Equal)	N2S ≠ W2E	N2S ≠ W2E

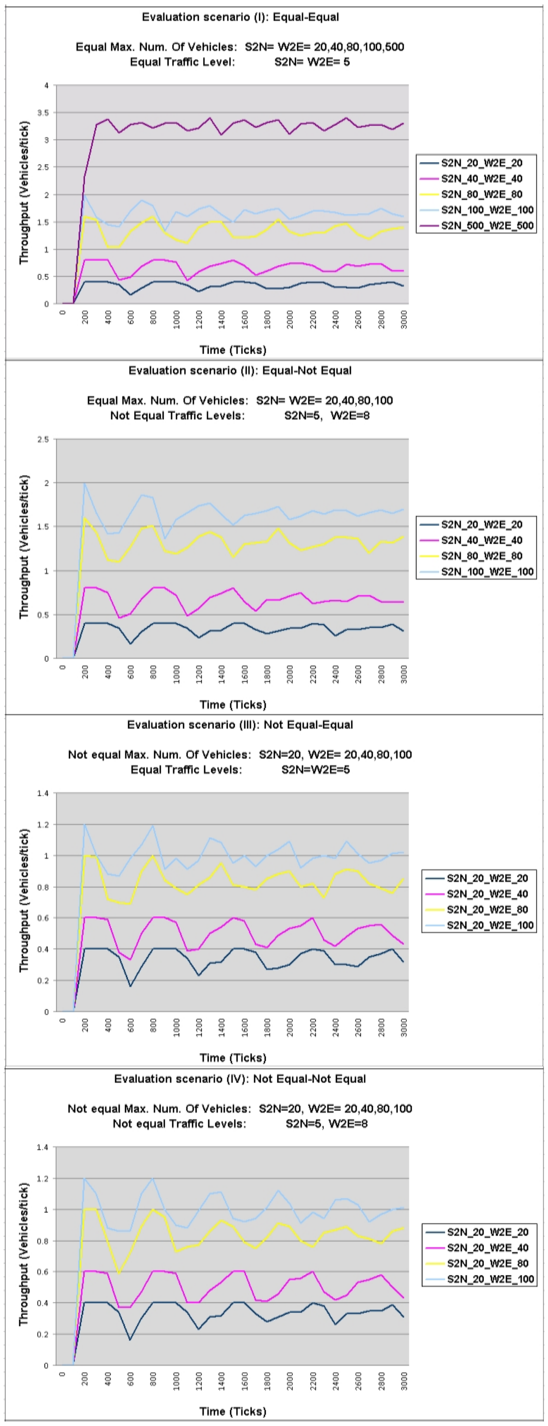

Figure 5. The system throughput (# Vehicles/tick) for each evaluation scenario (I, II, III, and IV) in an interval between 0 und 3000 ticks

It can be seen that from approximately the tick (120) the vehicles begin to leave the intersection, because at the beginning of the simulation the intersection was empty. Therefore, the system throughput in this interval [0-120] is zero. Thereafter, the system throughput increases always with time in the case of the cumulative system throughput (#Vehicles), or it is at its best (i.e., approximately constant) in the case of the throughput per time unit (#Vehicles/tick). This note applies to the four evaluation scenarios.

Figure 6 shows the system throughput comparing the four evaluation scenarios according to varying the value of the maximum number of vehicles in each direction after 3000 ticks including the extreme case. Here, on the x-axis is the maximum number of vehicles together in both directions $W2E$ and $S2N$.

In evaluation scenario I, the system throughput increases almost always linearly with the number of vehicles. In a similar manner, the same behaviour of the system throughput applies to the other evaluation scenarios. However, this behaviour of the system throughput will be changed only in the extreme case, (500-500) vehicles.

In the extreme case, the system performance achieves a value of around 9500 vehicles, because the maximum number of vehicles here is greater than the capacity of the intersection. Thus, it can be concluded that the system throughput within the capacity of the intersection increases almost always linearly with the number of vehicles.

In evaluation scenarios I and II, the values of the system throughput are approximately identical. This means that the maximum number of vehicles in each direction is relevant, not the traffic levels (traffic flow rates) of vehicles in each direction. The system achieves a throughput of around 5000 vehicles by 100 vehicles in every direction in both evaluation scenarios I and II (see Figure 6). A similar conclusion can be obtained when the values of the system throughput in evaluation scenarios III and IV are compared.

However, it is obvious that the values of the system throughput in evaluation scenario III are not similar to the values in the evaluation scenario I and II, because the total amount of vehicles in both directions in evaluation scenario III is less than the amount in evaluation scenarios I or II.

In general, it can be noted that the system throughput increases almost always linearly with the number of vehicles in all evaluation scenarios (I, II, III, and IV) as long as the maximum number of vehicles is not greater than the capacity of the intersection.

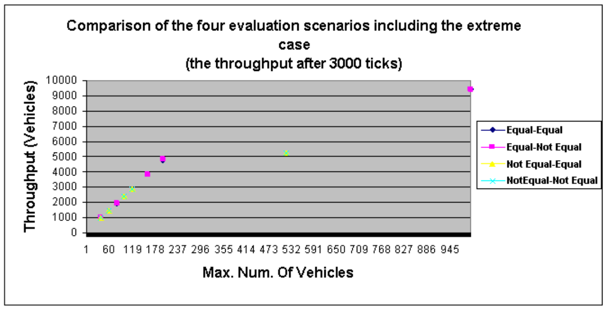

Figure 6. The throughput of system in the four evaluation scenarios including the extreme case after 3000 ticks

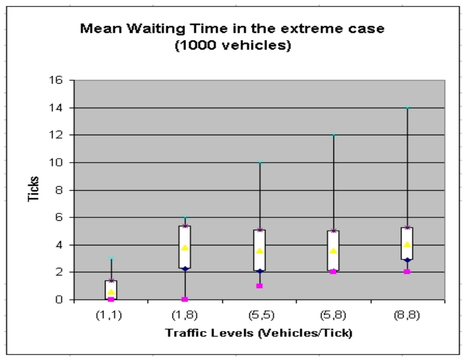

Figure 7. The mean waiting time in the extreme case (1000 vehicles)

C. *Results of mean waiting time measurement*

Here, the results of the mean waiting time metric will be discussed for the extreme case, 500 vehicles in every direction, where the maximum number of vehicles is greater than the capacity of the intersection (very busy intersection). That is done in order to show the longest waiting time at all with its standard deviation.

For this purpose, the measurements were repeated in the cases that the traffic levels of vehicles in south-north and west-east directions are: (1,1), (1,8), (5,5), (5,8), (8,8) vehicles/tick. The different rates at which vehicles enter each of the directions are chosen to investigate the effect of traffic streams with equal/unequal strength on the mean waiting time which vehicles experience as depicted in Figure 7 using a box plot. The mean waiting times and the standard deviations of all vehicles, that left the intersection, have been registered after 3000 ticks in the extreme case.

Despite the huge number of vehicles, which is greater than the capacity of the intersection, the resulting mean waiting times were low values with small standard deviations in all different traffic flow rates (traffic levels). The largest mean waiting time is by traffic rate (8,8) around $\Phi\ 4 \pm 1.19$.

V. CONCLUSIONS

In this paper, the path planning was the applied coordination mechanism to solve the problem of resource sharing wherever multiple agents (vehicles) cross the shared environment (centre of the intersection) avoiding collisions.

Path planning served to compute collision-free trajectories and to arrange the agents. The controller performs the path planning using a central planning algorithm and sends the planned trajectories to the agents only as recommendation. Here, a trajectory represents the path of an agent only inside the shared environment.

An adapted A*-algorithm for path planning of agents (vehicles) has been applied. The adaptation was necessary for the requirements of the used application scenario “intersection without traffic lights”, because a vehicle can only take a “rational” path. A*-algorithm searches the minimum-cost path in its three-dimensional configuration time-spaces. The implementation has been carried out under consideration of virtual obstacles that model blocked surfaces, restricted areas, which may arise as a result of reservations, accidents or other obstructions. The experiments showed a high performance of this algorithm. The evaluation of this algorithm was based on different test scenarios considering various loads of vehicles.

VI. FUTURE WORK

One aspect that may be of interest for future work is the fairness between the system’s agents (vehicles). In order to achieve this fairness, there are different approaches that deal with this issue. The other aspect that will be an important issue in the future is the coordination and cooperation of multiple intersections without traffic lights.

REFERENCES

[1] CAS-wiki: Organic Computing. http://wiki.cas-group.net/index.php?title=Organic_Computing, [retrieved: January, 2013].

[2] J.O. Kephart and D.M. Chess. “The vision of autonomic computing”. IEEE Comput. 1, 2003, pp. 41-50.

[3] R. Sterritt. “Autonomic Computing”. Innov. Syst. Softw. Eng. 1(1), 2005, pp. 79-88.

[4] Y. Chaaban, J. Hähner, and C. Müller-Schloer. “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, 2009, pp. 467-475, Greece. IEEE.

[5] Y. Chaaban, J. Hähner, and C. Müller-Schloer. “Towards Robust Hybrid Central/Self-organizing Multi-agent Systems”. In ICAART2010: proceedings of the Second International Conference on Agents and Artificial Intelligence, Volume 2, January 2010, pp. 341-346, Spain.

[6] Y. Chaaban, J. Hähner, and C. Müller-Schloer. “Handling of Deviations from Desired Behaviour in Hybrid Central/Self-Organising Multi-Agent Systems”. In Cognitive12: proceedings of the Fourth International Conference on Advanced Cognitive Technologies and Applications, July, 2012, pp. 122-128, France.

[7] Y. Chaaban, J. Hähner, and C. Müller-Schloer. “Measuring Robustness in Hybrid Central/Self-Organising Multi-Agent Systems”. In Cognitive12: proceedings of the Fourth International Conference on Advanced Cognitive Technologies & Applications, July 2012, pp. 133-138, France.

[8] R. Regele and P. Levi. “Cooperative Multi-Robot Path Planning by Heuristic Priority Adjustment”. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Peking, October 2006, pp. 5954-5959.

[9] M. Bennewitz and W. Burgard. An Experimental Comparison of Path Planning Techniques for Teams of Mobile Robots. In Autonome Mobile Systeme. Springer-Verlag, 2000.

[10] K. Dresner and P. Stone. “Multiagent Traffic Management: An Improved Intersection Control Mechanism”. In The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press, 2005, pp. 471-477.

[11] S. Leroy, J. P. Laumond, and T. Simeon. “Multiple Path Coordination for Mobile Robots: A Geometric Algorithm”. In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 1999, pp. 1118-1123.

[12] N.J. Nilsson. “Principles of Artificial Intelligence”, Springer Verlag, Berlin Heidelberg, 1982.