

# SHARED EXPERIENCES IN INTELLIGENT TRANSPORTATION SYSTEMS

Wilhelm Dangelmaier<sup>1</sup>, Holger Giese<sup>2</sup>, Florian Klein<sup>2</sup>,  
Hendrik Renken<sup>1</sup>, Peter Scheideler<sup>1</sup>

<sup>1</sup>Heinz Nixdorf Institut, University of Paderborn,  
Fuerstenallee 11, 33102 Paderborn, Germany

<sup>2</sup>Computer Science Institute, University of Paderborn,  
Warburger Str. 100, 33098 Paderborn, Germany

Abstract: In this paper, a scheme is outlined within which the autonomous vehicles in an intelligent transportation system exchange information about their environment. Benefiting from each other's experiences, the vehicles can locally adapt their behaviour in order to perform the desired flexible adjustments of the overall system behaviour. Based on the vehicles' shared knowledge, an exploration and planning strategy for selecting routes according to a set of given customer constraints is described. Meanwhile, sensor input from the environment is used to cooperatively learn the best routes and keep the knowledge about the world up-to-date. Copyright © 2004 IFAC

Keywords: Intelligent Transportation Systems, Exploration strategies, Communication of experiences, Function Approximation

## 1. INTRODUCTION

Intelligent transportation systems are characterized by automated computation, the demand for flexibility and freedom of choice for the autonomous vehicles, the demand for accurate, i.e. precise and up-to-date, information, a fundamental, enabling characteristic of transportation networks (e.g., in form of agents) and asynchronous, distributed algorithms for control, coordination, and resource management (Ghosh and Lee, 2000). Therefore, they can flexibly adjust their behaviour to the requirements supplied by the customers. In contrast, current rail systems are characterised by fixed timetables which cannot easily be modified once they have been elaborated by the operators. Even small localized changes may cause ripples of cascading delays to propagate.

In intelligent rail-based transportation systems, passengers can express their individual demands (starting point, destination) and preferences (routes, cost, travel time etc.). Based on these preferences, the system presents the passengers with customized offers. Consequently, transport services are provided in a demand-driven manner and not according to a central plan determined *a priori*.

Within the scope of the collaborative research centre "Self-optimizing Concepts and Structures in Mechanical Engineering"<sup>1</sup>, the Railcab Project<sup>2</sup> works on a passive track system and intelligent shuttles

transporting either goods or up to ten passengers. The shuttles operate individually and make independent and decentralized operational decisions. The project's vision is to combine the comforts of individual traffic, such as flexible scheduling, on-demand availability and customized cars, with the cost and resource effectiveness of public transportation. The self-optimisation additionally enabled the autonomous shuttles to observe changes of the environment and adjust their behaviour accordingly.

The relevant intelligent mechatronic systems consist of a multitude of individual components with a variety of tasks and functionalities. These components can be classified with respect to three hierarchical levels that support a modular decomposition of an overall problem into smaller sub-problems (Oberschelp et al., 2002; Gausemeier, 2002).

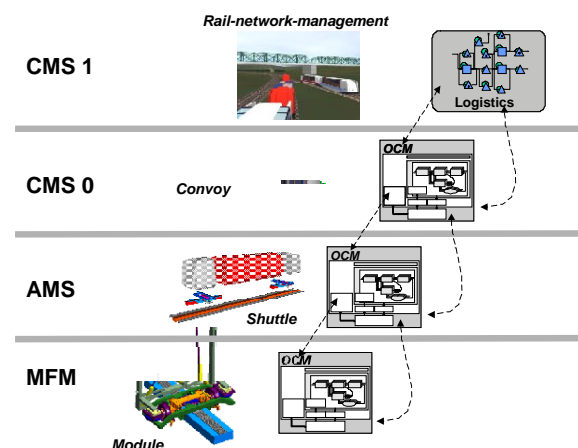


Fig. 1. Levels of intelligent mechatronic systems

<sup>1</sup> <http://www.sfb614.de>

<sup>2</sup> <http://www-nbp.upb.de/en/index.html>

The “Mechatronic Function Modules” (MFM) represent the lowest level. They basically consist of actuators, sensors, and the information processing facilities required for their operation. At a higher level of the hierarchy, “Autonomous Mechatronic Systems” (AMS) group various MFMs into larger units.

Functional processes assigned to multiple AMSs are represented on the highest level, “Cross-linked Mechatronic Systems” (CMS). As a CMS is defined by the informational integration between the involved AMSs, it does not imply a physical coupling, but the exchange and processing of information.

This paper concentrates on high-level CMS 1 dealing with logistics. It presents a pattern allowing multiple agents to co-operatively solve routing problems by sharing their knowledge (Wooldridge, 2003; Ferber 1999). It should be noted that in the context of the overall project the suggested solution is integrated with additional components, dealing with aspects such as safe operation. We therefore employ the techniques for a safety-critical real-time multi-agent system developed in (Giese et al., 2003a), which permits the compositional model checking of required real-time properties as outlined in (Giese et al., 2003b).

In Fig. 2, the relevant elements of the system and their relations are described by means of an UML class diagram. Based on this ontology, the requirements concerning the agents’ capabilities may be specified by means of interaction patterns, cultures and communities (Giese et al., 2003a).

In this paper, we will focus on an interaction pattern named “co-operative routing with learning”, which realizes a specific strategy for co-operatively finding the required routes while enabling collective learning through the sharing of information. Communicating their knowledge lets the shuttles benefit from other shuttles’ past experiences concerning fast or efficient routes, estimated travelling times and potential or actual obstacles such as traffic jams or accidents. Even though the setting is competitive, the co-operative behaviour is advantageous for both clients and shuttles and can therefore be considered rational. Under the assumption that only a limited number of shuttles (providers) are eligible for any particular task due to their current positions and schedules, contributing to the derivation of an optimal solution actually serves the self-interest of the large number of shuttles (advisors) unsuitable for the task at hand. They enjoy increased network efficiency without the risk of helping a direct competitor outbid themselves.

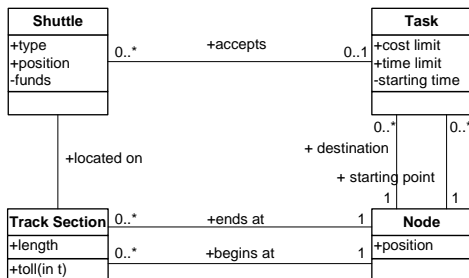


Fig. 2. Basic ontology of the system

This strategy is encoded by a set of norms called a *culture*. It defines the social context which requires its agent members to implement a set of interaction patterns and to fulfil the associated roles. In our example, we define a logistic culture that specifies that every member agent needs to be able to participate in the “co-operative routing with learning” pattern filling either the provider or advisor role.

Finally, communities are used as a concept to determine the agents’ concrete communication context at run-time. For each community, the requirements for membership and the culture it implements, defining the expected behaviour, are specified. In our example, the co-operation of the shuttles according to the logistics culture may either take place in a global community or be restricted to the shuttles of separate transportation companies.

The interactions that make up the “co-operative routing with learning” pattern are described in detail in the rest of the paper. Section 2 presents the procedure used by the shuttles to generate individual offers. In Section 3, the proposed co-operation process between the shuttles and the evaluation criteria for selecting an offer among the proposed routes are explained. Section 4 explains the employed feedback/learning mechanism. In the final sections, we review the related work and provide some concluding remarks.

## 2. CALCULATING AN INDIVIDUAL OFFER

### 2.1. Modelling passenger requests

We model the Railcab network as a directed graph  $G$ . Each vertex  $v_i$  corresponds to a terminal or track junction, whereas each edge  $e_j$  corresponds to a track section and may be annotated with a set of attributes. A path  $\pi$  is consequently defined as a sequence of connected track sections  $(e_i, e_j, \dots)$ .  $\Pi(G)$  denotes the set of all paths in  $G$ .

$$\Pi(G) \cong \left\{ (e_0, e_1, \dots, e_n) \left| \begin{array}{l} e_i \in E(G), i = 1..n \wedge \\ e_j.v_2 = e_{j+1}.v_1, j = 1..n-1 \end{array} \right. \right\} \quad (1)$$

Suppose a passenger desires to travel from a starting point  $s_c \in V$  to a destination  $f_c \in V$ . The passenger may additionally impose more specific constraints on the desired trip between these points, such as a desired starting time  $t_0$ , a factor  $\omega_c$  encoding the preference for either a fast or cheap trip, a limit  $d_c$  for the acceptable duration of the trip, a ceiling cost  $m_c$ , a set of preferences  $Type_c$  concerning the shuttle type (e.g. standard, luxury, cargo...) and special requests  $L_c$  imposing local acceptability restrictions on the parameters of individual track sections (e.g. no main lines, no tunnels, required minimum/acceptable maximum speed, ...). These requirements may be expressed abstractly as a constraint problem  $C$ :

$$C \equiv \{s_c, f_c, t_c, \omega_c, d_c, m_c, Type_c, L_c\} \quad (2)$$

The more restrictive the constraints defined by the passenger are, the smaller the feasible solution space

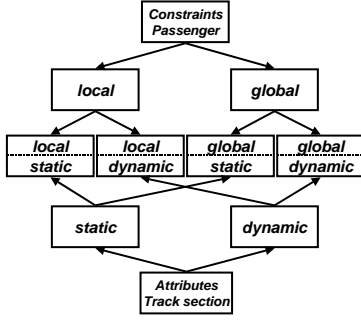


Fig. 3. Classification of different constraint types

will be. It is actually rather easy to inadvertently arrive at over-constrained problems.

When discussing attributes and the constraints that reference them, we differentiate between global and local constraints. These two classes can again be divided into static or dynamic constraints, yielding four distinct classes.

Local constraints pertain to the attributes directly associated with a single edge. A track that violates any of them may not be part of a feasible solution. Global constraints refer to attributes of a complete path  $\pi$ , such as the overall duration and cost ( $d_c$  and  $m_c$ ). They usually depend on the local attributes of the contained edges.

Static attributes do not change in the short term (e.g. the maximum allowed speed for a track section). Dynamic parameters do change frequently (e.g. the time required to traverse a track section, depending on traffic and weather conditions).

## 2.2. Reducing the graph

In the search for a valid route that fulfils all of the passenger's constraints, the shuttle (at least implicitly) needs to consider all possible paths from starting point  $s_c$  to destination  $f_c$ .

$$\Pi(s_c, g_c) \equiv \{\pi \mid \pi \in \Pi \wedge \pi.e_1.v_1 = s_c \wedge \pi.e_n.v_2 = f_c\} \quad (3)$$

For efficiency reasons, it is important to restrict the attention to those paths that may fulfil all constraints. Therefore, our method first uses a graph reduction algorithm which indirectly prunes away all paths violating the local constraints by removing all invalid track sections from the graph. Thus, the final path cannot possibly contain a track section that violates a local constraint.

FOREACH edge  $e_i \in E(G)$ :

  FOREACH local constraint  $l_c \in L_c$

    IF (edge  $e_i$  violates constraint  $l_c$ ) THEN

      remove edge  $e_i$  from Graph  $G$

## 2.3. Searching for the best route

The search for the best route requires an objective function that allows us to define a criterion for the evaluation of different paths. We define the cost of an

edge as the weighted sum of its monetary  $m_e$  and temporal  $d_e$  cost:

$$c_e = (1 - \omega_c) \cdot m_e + \omega_c \cdot d_e \quad (4)$$

As our objective, we try to minimize the total cost of the path given by the sum of the cost of the traversed edges:

$$c_\pi = \sum_{e \in \pi} c_e \quad (5)$$

In order to be able to estimate the cost of a path where no proper experiences exist, we define two approximation functions that try to infer probable values for  $m_e$  and  $d_e$  from known averaged coefficients.

The local attributes relevant to the monetary cost of traversing a track section are the average toll  $toll_a$ , a fixed-step cost for traversing it, the track's running length in kilometres  $length_e$ , the average cost per kilometre due to wear, expressed by the abrasion factor  $a_a$ , and the average energy cost per kilometre  $w_a$ . Together, these yield the estimated monetary cost function:

$$m_e^{est} = toll_a + (a_a + w_a) \cdot length_e \quad (6)$$

The speed  $|v_a|$  that shuttles on the network achieve on average leads to an estimate for the time expended when passing a track:

$$d_e^{est} = \frac{length_e}{|v_a|} \quad (7)$$

As these approximation functions provide rather coarse estimates, it is preferable to use cost functions based on actual experience wherever they are available.

The actual cost functions vary depending on the time of day and the day of the week to a great extent. Rush hour traffic will have an obvious adverse effect on  $d_e$ .

In order to steer traffic away from congested spots, the toll for busy segments might be increased by the traffic controllers. At the same time, denser traffic will have more subtle effects on energy consumption and wear due to a higher number of accelerations and decelerations. Therefore, the shuttles try to construct a profile for each track section representing the cost dependent on the time of the week. Such a profile consists of four functions, each spanning the 168 hours of a stereotypical week: The actual experience is encoded by the prediction functions  $m_e^{exp}(t)$  and  $d_e^{exp}(t)$ . The confidence in these predictions is encoded by the confidence functions  $cf_e^m(t)$  and  $cf_e^d(t)$ . The confidence functions basically derive from the number of supporting experiences that are available for a specific cost prediction and should therefore provide an indication of the quality of that prediction. The exact encoding of the functions and the algorithms used for their derivation are presented in detail in Section 4.

Alongside the confidence factors, additional buffer factors may be defined to account for the inescapable variations of the actual cost around the predicted value. As shuttles may incur contractual penalties when violating user-specified constraints, these factors may be adjusted depending on personal risk aversion,

current financial situation, the particulars of a specific order, and records of past performance regarding constraint satisfaction. These adjustments can be effected either directly by the shuttle's stakeholder or through a process self-optimization. Separate factors  $bf_m$  and  $bf_d$  are specified for monetary and temporal cost. The influence of the buffer factors is increased for lower confidence levels.

$$\lambda_m = 1 + bf_m (2 - cf_e^m(t)) \quad (8)$$

$$\lambda_d = 1 + bf_d (2 - cf_e^d(t)) \quad (9)$$

This leads to the following cost function:

$$c_e(t) = (1 - \omega_c) \cdot \lambda_m m_e^{\text{exp}}(t) + \omega_c \cdot \lambda_d d_e^{\text{exp}}(t) \quad (10)$$

For searching the best route, we use the A\*-algorithm (Russell and Norvig, 2003; Beckstein 2003). Originating from the source node  $s_c$ , it picks the unprocessed node with the currently best evaluation and then proceeds to consider the directly reachable successor nodes  $n'$  for possible improvements regarding their evaluation. The A\*-algorithm differs from Dijkstra's simpler shortest paths algorithm in the use of a heuristic that incorporates a lower bound on the cost for the distance remaining between a node and the destination into the evaluation of the node. This allows it to perform a depth first search directed at the destination and thus avoid visiting a possibly large number of nodes in a breadth-first search.

As the evaluation function, we use an estimate for the minimal cost of a path incorporating  $n$  and connecting source and destination. This consists of the exact cost of the best known path from the source to  $n$  and the heuristic yielding a lower bound on the cost for the remaining trip from  $n'$  to the destination. As the cost functions are dependent on the time of the week, the respective times of arrival  $t_{n'}$  for node  $n'$  are propagated and stored along with the current evaluation.

$$c^{A^*}(n', t_{s_c}) = c(n', t_{s_c}) + h(n', f_c, t_{n'}) \quad (11)$$

The cost for the best path up to a node  $c(n, t)$  is computed based on the cost functions introduced above and is given by the following recursive definition:

$$c(n', t) = c(n, t_{s_c}) + c_{(n, n')}(t_n) \quad (12)$$

The behaviour of A\* is strongly dependent on the heuristic  $h(n', f_c, t_{n'})$ . If the heuristic is not a correct lower bound on the cost of any real route between  $n'$  and  $f_c$ , the algorithm may terminate prematurely and yield a suboptimal solution. Therefore, we impose the following restriction on the heuristic, thus guaranteeing that A\* will always find the best possible path (Russell and Norvig 2003; Beckstein 2003):

$$h(n', f_c) \leq c_e \quad \forall \pi \in \Pi(n', f_c) \quad (13)$$

If, however, the heuristic provides overly optimistic estimates, thus yielding a bound that is not sufficiently tight, A\* degenerates into Dijkstra's algorithm and squanders any possible performance gains.

In the case of the shuttle routing problem, we can consider our graph as a two-dimensional geometric space, basically representing a map of the network. This suggests the use of a heuristic based on the Euclidian distance, which corresponds to the minimal possible

distance between two nodes (note that the bound is valid, as the incorporation of correct topographic information would tend to only increase the distance). In order to compute the actual cost, we need a bound on the lowest feasible costs per kilometre  $m_e^{\text{avg}}$  and  $d_e^{\text{avg}}$  for the relevant time of the week. This corresponds to finding the minimal values for  $m_e(t) / \text{length}_e$  and  $d_e(t) / \text{length}_e$  in the time window between  $t_{n'}$  and the latest admissible time of arrival  $t_c + d_c$  specified in the constraints, considering all edges. A definition for  $h(n', f_c, t_{n'})$  fulfilling the above constraint is therefore:

$$\Delta(n', f_c) = \sqrt{(x_{n'} - x_{f_c})^2 + (y_{n'} - y_{f_c})^2} \quad (14)$$

$$h(n', f_c, t_{n'}) = \Delta(n', f_c) \left[ (1 - \omega_c) \cdot \min_{[t_{n'}, t_c + d_c]}(m_e^{\text{avg}}) + \omega_c \cdot \min_{[t_{n'}, t_c + d_c]}(d_e^{\text{avg}}) \right] \quad (15)$$

It should be noted that the provided solution is only optimal w.r.t. the cost functions used by the shuttle, which will never provide an entirely accurate representation of the real conditions. A more serious problem arises, however, when the solution computed by A\* does not fulfil both global constraints.

#### 2.4. Considering Multiple Global Constraints

As finding a path which is subject two multiple additive constraints is NP-complete (Korkmaz et. al. 2000), we did not consider the global constraints when searching for the best route. As our evaluation criterion was based on a weighted sum of the monetary and temporal cost of a route, the following cases may arise when taking the global constraints back into consideration:

- If the solution provided by A\* respects the limits imposed by both global constraints, the solution is both optimal and feasible. Our rationale for the use of A\* is the assumption that, for appropriate values of  $m_c$ ,  $d_c$  and  $\omega_c$ , this will actually be the most frequent case.
- If the optimal solution violates both constraints, there is no feasible solution.
- If the solution violates only one constraint, there may or may not be a feasible solution.

The last case is the most complex. A simple, often successful strategy is to continue the enumeration of solutions by A\* (ignoring the found solutions which do not fulfil one constraint) and simply test whether they satisfy both limits. If this strategy is not successful, there are more sophisticated heuristics that first try to move the optimum into the feasible region of the solution space by adjusting  $\omega_c$  and then, if this does not result in a success, subsequently scale the constraint axes (cf. (Korkmaz et. al. 2000)). It is to be noted that these heuristically found solutions need not arrive at the optimal value for  $c$ .

### 3. ASKING FOR ADVICE

If the A\* algorithm yields a feasible route with a low confidence factor for which there is hardly any experience available to the shuttle, the shuttle asks other shuttles for a more reliable solution. This decision is made by comparing a path's accumulated confidence factor with a shuttle's confidence constant  $\alpha$ . The

confidence factor of a path is the minimal confidence factor for any of the edges in the path:

$$cf_{\pi}(t) = \min_{e \in \pi} cf_e(t_{e,v_i}) \quad (16)$$

If its confidence in its own solution is below  $\alpha$ , it asks other shuttles for advice.

The selection of the other shuttles is based on a similarity function. Each shuttle is characterized by a set of properties P:

Type = {passenger | goods}  
Year of construction = {1980, 1981, 1982 ...}  
Model = {Siemens X-Call, ABB Z5, ...}  
Sort = {Comfort | Racing | Standard, ...} etc.

Drawing from a pool of potential advisors offering their assistance via a service registry, the request for advice will be sent to a selection of advisors consisting of those most similar to the sender. The similarity is calculated using an appropriate similarity function. The shuttle transmits both the complete set of constraints  $C$  and the desired buffer factors. If an advisor intends to compete as a provider, it may reject the query. Otherwise, it computes an optimal path using the above procedure and returns it along with the associated confidence factor (Advisors might specialize, invest in more processing power and try to build more accurate track profiles by exchanging and accumulating experiences, selling reliable and informed solutions as a premium service).

After having received  $n$  solutions specifying a path, its expected cost  $c_i$  and the associated confidence factor  $cf_i$  from the advisors, the shuttle picks a solution based on the evaluation function  $ef_i$ :

$$ef_i^a = \left( \sum_{j=1}^n c_j / n c_i \right)^r \cdot cf_i^{\frac{1}{r}} \quad ef_i = ef_i^a / \sum_{j=1}^n ef_j^a \quad (17)$$

$ef_i$  formalizes the trade-off between lower cost and greater confidence, the first factor representing the relative attractiveness of the proposed cost and the second factor being the confidence factor. Thus, setting  $r > 1$  emphasizes cost, whereas  $r < 1$  shifts the emphasis on reliability.

Table 1 Evaluation of the solutions

Solution	$c$	$cf$	$ef$
S <sub>local</sub>	22	0,6	0,22
S <sub>1</sub>	28	0,5	0,15
S <sub>2</sub>	24	0,8	0,31
S <sub>3</sub>	32	0,9	0,29
S <sub>4</sub>	12	0,1	0,03

Shuttles might now rank the proposals by the value of  $ef_i$  and choose the most attractive alternative. Choosing deterministically has several disadvantages, however. If all shuttles chose small values for  $r$ , previously unknown edges might never be explored due to the shuttles' overly conservative preferences. Furthermore, it might lead to congestion on 'optimal' routes, making apparently suboptimal routes actually more advisable. The values for  $ef_i$  might therefore also be seen as the parameters of a probabilistic choice function. Table 1

shows an exemplary configuration. In the case of the most attractive solution  $S_2$ , the suboptimal cost is more than compensated by the higher confidence factor.

#### 4. LEARNING AND FEEDBACK

For each track-specific cost or duration profile a shuttle maintains, it needs to store two functions: the approximated profile  $a_i^p(t)$  and the support  $s_i^p(t)$ , a quality measure that is roughly based on the number of validating experiences supporting the estimation at a particular point of the profile. As an efficient and compact representation, we use a sufficiently dense grid of evenly spaced interpolation points and compute the missing values using cubic spline interpolation (Cohen, *et al.*, 2001; Knott, 2000). At moderate computational cost, cubic polynomials give an exact fit and avoid spurious oscillations.

The learning algorithm basically works by moving the approximation curve towards a new measurement  $x_k$  while incrementing the support value. Based on the assumption that the actual profile is not characterized by high frequency, high amplitude fluctuations, the new experience does not only affect the curves at the exact point of measurement  $t_k$ , but also in a neighborhood of size  $\tau$ . Its influence decreases with distance from  $t_k$ . The influence's strength is given by a weight  $\omega_j$  in the interval  $[0,1]$  that is determined by a Gaussian function:

$$\omega_j = e^{-\frac{9(t_j - t_k)^2}{2\tau^2}}, \text{ so that } \omega_j \rightarrow 0 \mid t \notin [t_k - \tau, t_k + \tau]. \quad (18)$$

$\tau$  itself is adaptively chosen so that the (approximated) definite integral of the support function over the interval  $[t_k - \tau, t_k + \tau]$  equals a constant  $S^p$ . The rationale behind this is that we want to use larger neighborhoods when there is little experience in order to adapt quickly, but use small neighborhoods that yield a more exact reproduction of a profile's details once the rough outline is sufficiently supported by experience – not unlike the concept of simulated annealing.

Measurements may be affected by random fluctuations. They therefore do not simply replace previous estimates, but gradually change them. The new estimate at each interpolation point is computed as a weighted sum, where  $\omega_j$  and the support function serve as the weights:

$$a_{i+1}^p(t_j) = (s_i^p(t_j) \cdot a_i^p(t_j) + \omega_j x_k) / (s_i^p(t_j) + \omega_j) \quad (19)$$

The support function is basically updated by adding  $\omega_j$  at each interpolation point. If the support function is allowed to grow in an uncontrolled manner, however, the approximation will soon become very resistant to change. As the actual profile is expected to change over time, there are two mechanisms accounting for the fact that experiences become outdated. Globally, the entire support function is multiplied by a degressive factor  $\lambda$  at the end of each period. Locally, the support value is reduced whenever the difference  $\Delta$  between the current estimate and the new measurement is larger than some threshold  $\phi$ . The rationale here is that contradictory experiences do not increase confidence, but actually introduce uncertainty. The strength of the effect depends on  $\Delta$  and the two parameters  $\kappa$  and  $\rho$ :

$$s_{i+1}^p(t_j) = (s_i^p(t_j) + \omega_j) \min\left(1, \left(\kappa + (1 - \kappa)e^{\rho^2 - (\rho\Delta/\phi)^2}\right)\right) \quad (20)$$

When using the profile for routing, the confidence function  $cf$  is derived directly from the support function. The conversion function scales the support value down to  $[0,1]$ . It assumes a diminishing utility of additional experiences, but asymptotically approaches 1 rather quickly, as support is generally expected to be sparse.

$$m_e^{\text{exp}}(t) = a_i^{(m,e)}(t) \quad (21)$$

$$cf_e^m(t_i) = 1 - \gamma^{s_i^{(m,e)}(t_i)} \quad | \quad 0 < \gamma \leq 1 \quad (22)$$

By modifying  $\gamma$ , the number of experiences required for a high degree of confidence can be adjusted.

## 5. RELATED WORK

The realization that the efficiency of a transportation network is strongly affected by the information flows that complement the physical processes has spawned an ever increasing interest in intelligent transportation systems. Currently, there is a trend away from centralized towards distributed solutions (Ghosh and Lee, 2000). While our approach shares conceptual elements with various other proposals, the shuttles' revolutionary concept and complete autonomy leading to the notion of a multi-agent system (Wooldridge, 2003; Ferber 1999) set it apart and motivate our original contributions.

As the multiple constraint path selection problem (MCP) is already NP-complete (Korkmaz et. al., 2000), a heuristic solution has to be employed for the effective analysis of the required routes. For the case restricted to only two additive constraints, we can employ the heuristics proposed in (Korkmaz et. al., 2000). If more than two additive constraints are relevant, heuristics for multiple constraints, which for example work with restricted propagation sets as presented in (Yuan 2002), might be used. Another alternative are genetic algorithms for multi-objective optimization as outlined in (Coello, 2000).

The presented approach is currently restricted to the case of inflexible orders. In practice, customers may readjust their plans and needs, which requires dynamic re-planning. An extension of the A\*-algorithm for re-routing plans as proposed in (Stenz et. al.; 2002) can be employed in this case.

## 6. CONCLUSION

We have shown an interaction pattern for a logistic multi-agent system in which heterogeneous agents plan their actions and explore the environment by using their own experiences as well as those of other agents. We are currently implementing a simulation environment that will allow us to test our assumption that co-operation between the agents generates better results than autonomous searches solely based on proper experiences.

## ACKNOWLEDGEMENTS

This work was developed in the course of the Collaborative Research Centre 614 – Self Optimising Concepts and Structures in Mechanical Engineering – University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

## REFERENCES

- Beckstein, C. (2003). Suche. In: *Handbuch der Künstlichen Intelligenz* (Görz, G., C.R. Rollinger and J. Schneeberger (ed.)), Oldenbourg Verlag, München.
- Coello Coello, C. A. (2000). An Updated Survey of GA-Based Multiobjective Optimization Techniques. In: *ACM Computing Surveys, Vol. 32, No. 2, June 2000*.
- Cohen, E., R.F. Riesenfeld and G. Elber (2001). *Geometric Modeling with Splines*, pp. 43-67. A K Peters, LTD, Natick, USA..
- Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, pp. 59-85. Addison-Wesley, Essex, England.
- Gausemeier, J (2002). Von der Mechanik zur Selbstoptimierung. *20th CAD-FEM Users Meeting*, Friedrichshafen, Germany.
- Ghosh, S.; Lee, T. (2000): *Intelligent Transportation Systems. New Principles and Architectures*. CRC Press, Boca Raton.
- Giese, H., Burmester, S., Klein, F., Schilling, D., Tichy, M. (2003a). Multi-Agent System Design for Safety-Critical Self-Optimizing Mechatronic Systems with UML. In: *Proceedings of the 2<sup>nd</sup> Workshop on Agent-Oriented Methodologies, OOPSLA, Anaheim, USA*.
- Giese, H., Tichy, M., Burmester, S., Schäfer, W., Flake, S. (2003b). Towards the Compositional Verification of Real-Time UML Designs. In: *Proc. of the European Software Engineering Conference (ESEC), Helsinki, Finland*.
- Knott, G.D. (2000). *Interpolating Cubic Splines*, pp. 63-75. Birkhäuser, Boston.
- Korkmaz, T.; Krunz, M.; Tragoudas, S. (2000). An efficient algorithm for finding a path subject to two additive constraints. In: *Proceedings of the 2000 ACM SIGMETRICS*, Santa Clara, CA, USA.
- Korkmaz, T.; Krunz, M. (2003). Bandwidth-Delay Constrained Path Selection Under Inaccurate State Information. In: *IEEE/ACM TRANSACTIONS ON NETWORKING*, Vol. 11, No. 3, pp. 284-398, June 2003.
- Luger, G.F. and W.A. Stubblefield (1993). *Artificial Intelligence. Structures and Strategies for Complex Problem Solving*, pp. 75-114. The Benjamin/Cummings Publishing Company, Inc. Redwood City, California.
- Oberschelp, O., T. Hestermeyer, B. Kleinjohann and L. Kleinjohann. (2002). Design of Self-Optimizing Agent-Based Controllers. In: *Workshop 2002: Agent Based Simulation*, (Christoph Urban (ed.)), SCS European Publishing House, Erlangen, Ghent.
- Russell, S.J. and P. Norvig (2003). *Artificial Intelligence: A modern Approach*, pp 59-89. Pearson Education International, Upper Saddle River, New Jersey.
- Stenz, A. (2002). CD\*: A Real-time Optimal Re-planner for Globally Constrained Problems. In: *Proceedings of AAAI 2002*, July 2002.
- Wooldridge, M. (2003). *An Introduction to Multiagent Systems*, pp. 189-218. Jon Wiley & Sons, LTD, West Sussex, England.
- Yuan, X. (2002). Heuristic Algorithms for Multiconstrained Quality-of-Service Routing. In: *IEEE/ACM TRANSACTIONS ON NETWORKING*, Vol. 10, No. 2, pp. 244-256, April 2002.