Agent-Based Optimization of Maintenance Planning for Railway Vehicles

Julian Franzen[1], Udo Pinders[2] and Bernd Kuhlenkötter[1]

[1]*Chair of Production Systems, Ruhr-University Bochum, Bochum, Germany.*
franzen@lps.rub.de,
kuhlenkoetter@lps.rub.de
[2]*Westfälische Lokomotiv-Fabrik Reuschling GmbH & Co. KG, Hattingen, Germany.*
u.pinders@reuschling.de

ABSTRACT

In practice, maintenance of rail vehicles is based on reactive and preventive maintenance strategies. Condition-oriented maintenance approaches are only slowly finding their way into the market. When researching the state of the art, it is noticeable that the majority of the approaches presented is considering individual components - the system focus necessary for maintenance optimization is not taken into account.

Depending on the target system (number of components) and planning period, a complex optimization problem (OP) results. The OP is an NP-heavy problem for which the use of Genetic Algortihms can deliver suitable solutions for small search spaces. When applying it on a complex system with a larger solution space, this heuristical approach alone is not sufficient for the analytical optimization of a system representing a locomotive.

Therefore, in this paper agent-based distributed problem solving is applied to analytically optimize the maintenance of the target system. Therefore, a multi-agent system (MAS) based on the O-MaSE-model will be developed, which captures the configuration of a target system and formulates the overall OP using the fictional data from a drivetrain of a shunting locomotive as an example. Following the principle of co-evolutionary problem solving, the overall problem is divided into smaller subproblems (SP). These SP have the right size to be solved by an own agent using genetic algorithms. In addition to, the solution focuses on the autonomous negotiation of an acceptable solution for the entire system by the SP agents.

1. MAINTENANCE OF SHUNTING VEHICLES

Freight transport is increasingly competing with other modes of transport, especially road transport. A disadvantage in this context are the high life cycle costs (LCC) of locomotives, which are required for regular operation and shunting, i.e. the composition of a train. A large part of these life-cycle costs results not only from operation, but from maintenance.

Shunting locomotives, like the majority of all rail vehicles, are maintained on time-based intervals. The vehicle represents a system of individual components. The components can have one or more different failure modes. In practice, inspections, maintenance and repairs are combined in so-called maintenance levels. The underlying maintenance strategies are reactive and preventive. Due to frequently discussed disadvantages, economic and ecological disadvantages result from these strategies (Swanson (2001)).

Predictive maintenance has developed in response to this. It has still a low level of maturity and is therefore mainly subject of research, whereas only small real application scenarios exist.

As one of several strategies, Prognostics- and Health Management (PHM) has established itself as a methodological support for predictive maintenance. (Goodman et al. (2019)) The PHM-model can be divided into two areas. The first area deals with the collection and analysis of data in order to determine conditions. Most of research can be found in this area. Usually, the focus is set on isolated components. Frequently considered components are, for example, pantographs, couplings and wheelset bearings. Atamuradov (2017) provides a good cross-industry overview.

The second area of the model focusses on decision making. Up to now, the focus has been little on the transfer into operational processes, also due to a still small but growing number of use cases. Therefore, in this paper a multi-agent system (MAS) for the coordination of condition-based maintenance of a system consisting of individual components will be designed and exemplarily applied. The MAS will act as a decision support system (DSS) for maintenance planners of condition-based maintained systems.

For this purpose, maintenance planning is initially described as an optimization problem. It is shown that this problem represents a typical application case for heuristics, but the complexity requires a distributed problem solution. Then a MAS for the realization of distributed problem solution using the O-MaSE model is presented. This paper concludes with the implementation and simulation-based testing of the distributed problem solution.

2. FORMULATION OF THE OPTIMIZATION PROBLEM

Various types of optimization problems are defined in the literature. The analytical maintenance optimization of a rail vehicle as a system of components can be classified as a periodic maintenance scheduling problem (PMSP). The optimization goal in the context of predictive maintenance is a cost and availability optimized maintenance plan for the target system rail vehicle, which is abstracted by a system of single components. In the following, the problem representation, the objective function and the constraints relevant to the optimization problem are introduced. The description is based on the state of the art for PSMP (e.g. Mansour (2011)). Finally, the role of the genetic algorithm in solving PSMP is explained.

2.1. Problem representation

The maintenance planning is realized by a binary problem representation X. A rail vehicle has n components whose maintenance has to be planned over a time horizon of several years. This time horizon can be divided into $steps$ time steps. The duration of a time step is $timestep$ hours. A component can be maintained at any time t ($X(t) = 1$) or not ($X(t) = 0$). Figure 1 outlines the problem representation X as input to the objective function. The format of X is $n \times steps$ (see Figure 1).

$$X = \begin{matrix} component_1 \\ component_2 \\ \dots \\ component_n \end{matrix} \begin{bmatrix} 000010000100001 \\ 000000010000001 \\ \dots \\ 100010001000010 \end{bmatrix}$$
$$t_1\ t_2 \qquad \dots \qquad t_{steps}$$

Figure 1. Binary problem representation

The objective function of the optimization problem uses the input variable X (maintenance strategy) to determine the costs associated with the optimization problem (Eq. (1)):

$$f(X) = cost(X) \tag{1}$$

2.2. Cost function

To determine the costs $Cost$ for the maintenance strategy to be valuated, the system first considers which component is being repaired at what time. This maintenance results in costs $Cost_M$ from the repair or replacement of the component (C_{Comp}) and the work to be performed (C_{Work}) according to equation (2).

$$Cost_M(X) = C_{Comp}(X) + C_{Work}(X) \tag{2}$$

However, a correction is necessary, as synergy effects between joint maintenance of several components are not taken into account. An example is the parallel maintenance of axles and wheel discs. For both maintenance actions, the same preparatory work (jacking up the locomotive, removing the wheel sets, pressing off the components, etc.) is required, which involves a not inconsiderable amount of work. If maintenance effort is only considered for each component, then work that occurs only once is considered several times.

In this approach, differing from state of the art, availability is taken into account via costs. For this purpose, the downtime is determined and the cost of the replacement service is calculated for the downtime. These costs result from the temporary rental of a replacement vehicle, for example. In this way, the downtime and thus the availability are converted into a cost factor $Cost_{DT}$ and taken into account according to Eq. (3).

$$Cost_{DT}(X) = Downtime * C_{replace}(X) \tag{3}$$

The total costs are then calculated according to Eq. (4).

$$Cost = C_M + C_{DT} \tag{4}$$

2.3. Constraints

The constraints of an optimization problem limit the space of valid solutions. For the coordination of maintenance, the constraints result from the wear and failure behavior of the considered components.

In the past, the description of the failure behaviour of technical components and systems has been established by the Weibull distribution. This paper also describes the failure behaviour by the Weibull curve. Eq. (5) and (6) show the probability of survival R and the probability of failure G of a component

$$R(t) = e^{-(\frac{t}{T})^b} \tag{5}$$
$$T: characteristic\ lifetime,\quad b: form\ parameter$$

$$G(t) = 1 - R(t) \tag{6}$$

The constraints result from a limitation of the acceptable failure probability of the respective component. This can be set at a general limit value G_{max}, or assume an individual value for each component based on the risk tolerance of the respective vehicle operator. Due to the exponential terms in equations (5) and (6), there are non-linear conditions c which result according to equation (7).

$$c = G(t) - G_{max}\ !< \ 0 \tag{7}$$

2.4. Genetic algorithm for problem solving

Optimization problems are divided into classes according to their complexity. Like other well-known problem types like the Traveller Problem or the Knapsack Problem, scheduling problems belong to the class of NP-hard problems (Grigoriev et al. (2006)).

The objective function presented here is discontinuous and has many local minima. A very large configuration set of a size of $2^{n*steps}$ makes it difficult to solve the problem by exhaustive methods. In literature, heuristics have proven to be suitable for solving such problems. Heuristics are mathematical methods which help to find acceptable solutions for a problem in a short time using the nature-inspired principle of evaluation. Prominent examples are genetic algorithms (GA), ant algorithms and simulated annealing. A problem of the application of heuristics is that no statement can be made as to whether the solution found is a local or global minimum (see also validation problem, Fischer and Kruschwitz (1981)).

In the literature, the GA was established as a suitable approach to solve PMSP. For small configuration sets this approach provides good results, but with an increasing number of optimization variables the ability of the GA decreases significantly. Yang et al. (2008) estimate the limit at about 500 optimization variables for general scheduling problems, Franzen et al. find that the limit of applicability for PMSP is in a lower range. Therefore, a MAS for distributed problem solving will be designed and exemplarily applied in the following. For the evaluation of the solution, the criteria for heuristic solution evaluation according to Kirsch (1973) are applied.

3. DISTRIBUTED PROBLEM SOLVING AND MULTI-AGENT-SYSTEMS

In this chapter an introduction to multi-agent systems and the concept of distributed problem solving, especially co-evolutionary problem solving, is presented.

3.1. Definition of multi-agent systems (MAS)

Prior to the conception of the MAS, a short definition of the MAS will be introduced and the application field of distributed problem solving will be examined. According to VDI 2653-Part 1, (multi-)agent systems are "a set of agents that interact to perform one or more tasks. An agent is "a delimitable hardware and/or software unit with defined targets". An agent is designed to achieve these goals through autonomous behavior, interacting with its environment and other agents. This is the main difference between agent orientation in software development (AOSD) and conventional object-oriented software development (OOSD).

Agents are a modelling concept for solving technical, but also organisational and information technology tasks and are independent of a specific form of realisation. Parallel to object-oriented software development there are methods to develop MAS. An overview of existing methods is provided by VDI 2653-Part 2 (2018).

3.2. Introduction to distributed problem solving

The centralized approach of solving a complex problem by a single agent is often not effective. Analogous to classical software development, the task can significantly exceed the capabilities and resources of the agent. (Sycara et al. (1996)) The approach is therefore to solve a complex problem in MAS by dividing tasks among specialized agents, let them communicate and cooperate to realize distributed problem solving. Well-known approaches are Multidisciplinary Design Optimization, Distributed Constraint Optimization and co-evolutionary optimization.

For the solution of PMSP, co-evolutionary optimization has proven to be a promising approach. Beyond the evolutionary solution of problems, co-evolutionary problem solving interprets the optimization problem as an ecosystem in which different species live. These species adapt to their environment via evolutionary methods, but take into account the existence of representatives (individuums) of other species. (Potter and Young (2000)) The representatives can get chosen randomly or e.g. by their fitness value.

To apply co-evolutionary problem solving, the overall problem is divided into sub-problems, which form the species. The solution of the sub-problems is carried out taking into account the solutions of the other sub-problems (representatives). The overall solution finally forms the permissible combination of interdependent individuals who have in sum the best fitness (see Figure 2).

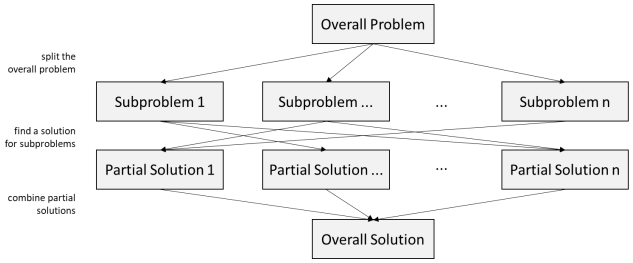

Figure 2. Distributed problem solution (Brenner, 1998)

4. CONCEPTION OF THE AGENT SYSTEM

Many approaches like Gaia, SODA and O-MaSE are available for the development of agent systems. The O-MaSE model is used for the development of the MAS in this paper, because it accompanies the development in all phases, can be used independently of the application language and platform and has already proven its suitability in several scenarios. Figure 3 shows the basic elements of a MAS organization in the O-MaSE metamodel. It becomes clear that the model documents individual activities and tasks during development, but deliberately does not document

(chronological) phases. In this paper, the development was carried out according to the Waterfall Model. In the following, requirements and roles of the MAS are identified. Then, the roles behaviours to realize co-evolutional optimization are described.

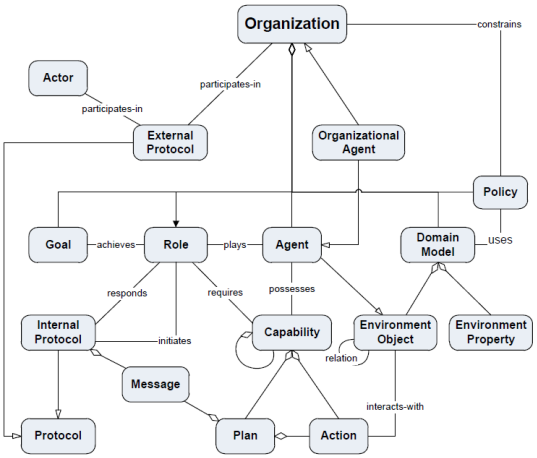

Figure 3. O-MaSE-Metamodel (DeLoach & Garcia, 2014)

4.1. Requirements for MAS

The requirements for the agent system result primarily from the formulation of the scientific question: How can condition-based maintenance of a rail vehicle as a system of components be planned at optimal costs and availability? A refinement of the objectives of the agent system by the development of use cases allows the establishment of a list of requirements and consequently the Goal Hierarchy Model. Figure 4 shows an exemplary use case diagram for requirements generation.

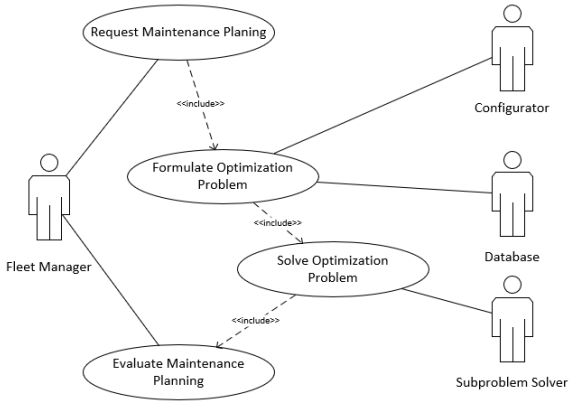

Figure 4. Exemplarily Use Case Diagram

Figure 5 shows the Goal Hierarchy Model for the MAS.

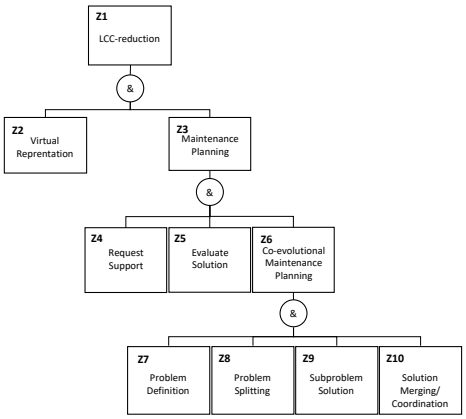

Figure 5. Goal Hierarchy Diagram

Looking at Figure 4, not only requirements or goals can be formulated from the diagram, but also a first estimation of involved roles and required activities can be identified. Therefore, the role diagram for the MAS can be derived (see Figure 6). From this, it becomes clear which roles are needed to fulfil the requirements and which communications take place between the roles.

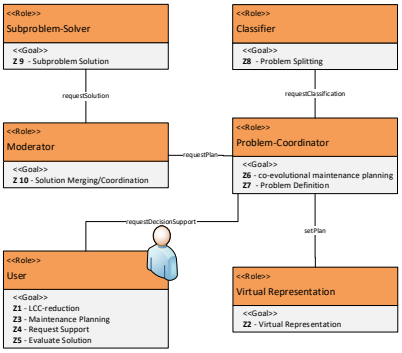

Figure 6. Role Model

In the following the development of distributed problem solving will be discussed.

4.2. Role Descriptions

After the roles have been identified, they are further specified in this section.

4.2.1. Problem-Coordinator

The need to instantiate an analytical maintenance optimization is communicated by the maintenance manager within the requestDecisionSupport communication.

If a problem solution is instantiated, the problem coordinator compiles the overall problem. The problem coordinator obtains the data from access to the virtual representation of the vehicle within the framework of the communication setPlan.

The problem coordinator also aims to support the process of distributed problem solving by providing an auxiliary process for organizing communication in the form of a blackboard. A blackboard is a common workspace in the MAS where agents can exchange information, knowledge and data. According to the classification of VDI 2653, the information comprises input data (COP), internal simulation data (partial problems and solutions for co-evolution, configuration data) and result data (overall solution).

Figure 7 summarizes the principle of the simulation blackboard (SBB), which is based on the schema of Brenner et al. (1998). The roles involved in the problem-solving process are described in the following subsections. The problem-solving coordinator only comes back into action after the distributed problem solution has been run through for the purpose of communicating results.

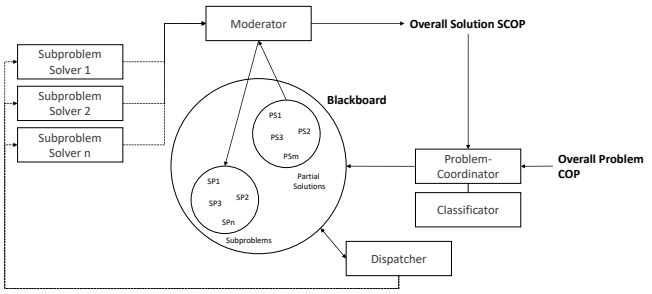

Figure 7. Simulation Blackboard

4.2.2. Classifier

The procedure for breaking down the overall problem into smaller partial problems of lower complexity includes the evaluation of the properties of the components and sorting by evaluation. The list sorted in this way is broken down into $nrSubProblems$ subproblems of the same size. The Agent *Classifier* role returns the number of subproblems and its components.

4.2.3. Subproblem-Solver

The *Subproblem-Solver* is responsible for solving the sub-problems. For this purpose, the role creates a sub-problem instance based on the parameters of the sub-problem and boundary conditions and solves it using the genetic algorithm. The problem representation, the objective function and the constraints correspond to the specifications according to section 2.

4.2.4. Moderator

With the previous roles, a mechanism was created to solve sub-problems. In this section, the development of the Moderator-role to develop a coordinated overall solution co-evolutionarily is shown. The coordination task requires the definition of the negotiation communication and strategy to force a compromise between at least two competing role objectives in order to combine partial solutions into an overall solution.

Co-evolutional Strategy

For the development of an overall solution, the solution environment is interpreted as an ecosystem in which the sub-problems represent individual species. Each species has a population which exists together with the population of the other species in the ecosystem. The adaptation to the ecosystem is done by using the genetic algorithm. When evaluating individuals, representatives of the other populations are taken into account as a fundamental difference to the conventional genetic algorithm. There are several possibilities for the selection of the representatives. These are either the best individuals or randomly chosen representatives.

In order to achieve the goal of a co-evolutionary solution to the complete optimization problem COP, the strategy of the moderator roles provides an iterative procedure, which starts with the creation of the initial representatives. Within the framework of the procedure described here, these are the best individuals of the respective population. These are requested and saved by the dispatcher after the partial problem solvers have been instantiated.

Subsequently, interdependent compromise solutions for the remaining subproblems are determined, taking into account the best representations found in the initial run. This results in step-by-step solution paths, which can be displayed in a solution tree.

In the following section the communication as an essential part of co-evolutionary optimization will be explained.

Co-evolutional Communication

The central object of communication is the communication of the representative from another species. As a way to transport this information, an auxiliary component is introduced, which is explained below.

Each individual of a population has a binary representation of the format $n \times steps$. Since steps can be assumed to be constant, the variation of the size of the search space is done via the parameter n, which represents the number of components considered in this problem. The variable n has to be chosen in such a way that a good solution can be found using GA. One possibility for the communication of the representative would be to combine the representative with an individual of the considered species and to evaluate its fitness. According to this, a problem representation of the format $(2 * n) \times steps$ results, which overstrains the GA. Therefore, the information of the considered representation is packed to a single component of the format $1 \times steps$, which is used as auxiliary component HK for co-evolutionary problem solving. Figure 10 illustrates the procedure for dimension reduction.

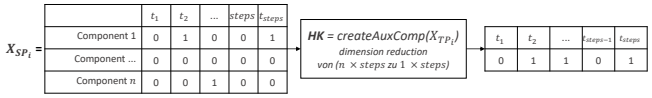

Figure 10. Dimension Reduction

The problem representation is then composed of the actual variables of the original subproblem to be optimized and the auxiliary component as representative of the species to be considered. While the GA optimizes the three original components, the auxiliary component remains static and acts as a fixed boundary condition. Thus, the genetic algorithm is forced to find a solution that implies a compromise.

Selection of a solution

The strategy is to create the solution tree and to identify the optimal solution path on the basis of this tree. In the case of the hierarchical organization intended for the agent system, the moderator does not leave the decision of a compromise to the agents involved, but chooses the most cost-effective variant in total and combines it in an overall solution vector.

5. IMPLEMENTATION

The subject of this chapter is the implementation of the agent system design developed in section 5 in a software prototype. According to the O-MaSE methodology for the development of organization-based multi-agent systems, which already accompanied the design of the multi-agent system in the previous chapter, this section corresponds to the subsequent environment-specific implementation.

5.1. Implementation environment

In this subsection, the implementation environment for simulation of the MAS will be described first. There is a variety of environments and frameworks for programming and subsequent simulation of MAS. Weyns and Michel (2015) give a broad overview of available solutions. For the implementation of the software prototype, the software AnyLogic turned out to be a suitable tool after comparing the requirements of the solution. The essential requirements for the software were the features of an agent-based simulation possibility, the plan-based implementation suitable for the use of the O-MaSE agent development model as well as the extensibility of the tool, which is fully guaranteed and offers maximum flexibility through the implementation of AnyLogic based on the Java programming language.

The heuristic algorithms for problem solving are implemented in MATLAB using the Global Optimization Toolbox. This toolbox provides powerful solvers for the application of heuristics, especially genetic algorithms. It offers the possibility to easily influence method-specific solver options such as mutation and inheritance mechanisms and thus fit them for the problem. The MATLAB software is connected to the AnyLogic simulation environment via the MATLAB Engine API to call the MATLAB scripts and functions containing the algorithms.

The implementation of the Blackboard as an exchange medium for the coordination of the overall solution and the subsequent provision is carried out on a MySQL database. The Java-based agent system is connected to the database via JDBC driver.

5.2. Implementation of the software prototype

In software development, prototypes are a way to obtain executable models at an early stage and use them for communication purposes. For this purpose, roles are first assigned to agent classes. The result is shown in Figure 11. The implementation of the four agent classes is done plan-based in AnyLogic, the implementation of the methods and functions in Java and MATLAB. Verification actions were performed continuously during the implementation.

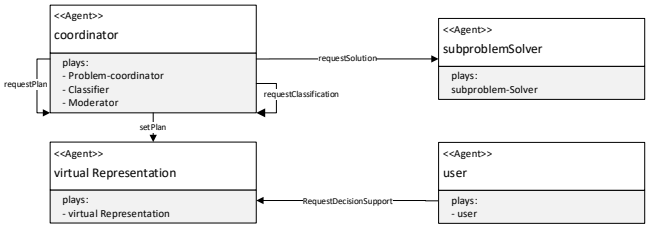

Figure 11. Agent Class Model

6. DEMONSTRATION

A simulative validation of the software prototype is carried out to demonstrate credibility of the MAS. According to VDI 3633 (2014), the validation of a system means "the verification of sufficient correspondence between model and original. It must be ensured that the model reflects the behavior of the real system accurately enough and without errors (Is it the correct model for the task?)". It should be emphasized that validation can only be formally operated to a certain degree and is often based on subjective assessments. A validated system is therefore not a formal proof of its correctness, but rather a proof that validation activities have been carried out. In this paper, validation aims not to provide formal proof of the validity of the model and its assumptions, but to confirm the credibility of the model. For this purpose, the considered target system is first transferred into a virtual representation in order to subsequently evaluate the function of the MAS on the basis of a validation scenario. It should be emphasized again that in the following, no validation is carried out using a real scenario, but the simulation is based on assumptions because real data is not available. However, the assumptions were not freely invented, but were extracted from historical data together with maintenance experts in an effort to achieve the greatest possible objectivity.

6.1. Transfer of the target system into its virtual representation

The subject of the study is the drive train as a section of the system of a diesel-hydraulic locomotive for shunting operation. As already described, valid failure data does not exist. In order to assume realistic values, maintenance documents of a rail vehicle maintenance company have been analysed and validated by means of expert interviews. Similarly, the costs have been determined in the same way. In order to protect company secrets, the values have been alienated. The components taken into account and the properties on which they are based are shown in Appendix 1 for reasons of space. To determine the bounds, a risk profile G_{max} was determined in consultation with maintenance experts. G_{max} includes permissible probabilities of 25 to 45 %.

6.2. Validation scenario

The scenario under consideration comprises the initial coordination of the condition-based maintained system at time $t = 0$. The result of the planning based on Table 1 is used to demonstrate the basic function of MAS. The subproblem solutions are evaluated under consideration of the optimal subproblem size and the negotiation mechanism.

6.2.1. Quality of subproblem solvers

For the evaluation of the solutions, a test plan with $N = 200$ samples for subproblem solutions of $subOpSize = 2 \dots 5$ is run through.

Quality of solution and optimal subproblem size

As mentioned at the beginning, on the one hand the quality of the solution is used as an evaluation criterion for the solutions found for the different subproblems. Since corresponding reference procedures for the reliable identification of the global minimum are missing, alternative qualitative characteristics are used for the evaluation of the solution, which are determined by manual consideration of the solutions:

- Criterion 1: Utilisation of reserves. It is checked whether the failure probabilities G do not exceed the limit value G_{max}, on the other hand they approach this value as close as possible before scheduled maintenance.
- Criterion 2: Use of synergy effects. It is checked whether the advantage of combining planned maintenance has been taken into account in the solution.
- Criterion 3: Verification that the heuristics have not planned maintenance that is not required according to the constraints.

Table 2 summarizes the results of this consideration for the above mentioned test.

Table 2. Qualitative evaluation of the solution quality for $subOpSize = 2 \dots 5$

$subOpSize = n$	Optimization variables $n * steps$	Criterion 1	Criterion 2	Criterion 3
2	72	✓✓✓	✓✓✓	✓✓✓
3	108	✓✓	✓✓	✓✓✓
4	144	✗	✗	✗
5	180	✗	✗	✗

Table 2 shows that under the given conditions a subproblem size of three components represents the limit of size.

Probability of solution

For the final evaluation of the results, the solution distribution according to Figure 12 for the subproblem size $subOpSize = 2 \dots 4$. Since the consideration of a different number of components entails higher costs and thus an absolute comparability is not given, only the relative distribution between the respective minimum and maximum value of the solutions belonging to the sample is considered. It becomes clear that the implemented genetic algorithm for $subOpSize = 3$ will most likely find a very good solution within known solutions. From $subOpSize = 4$ on, the probability of finding good solutions decreases rapidly.

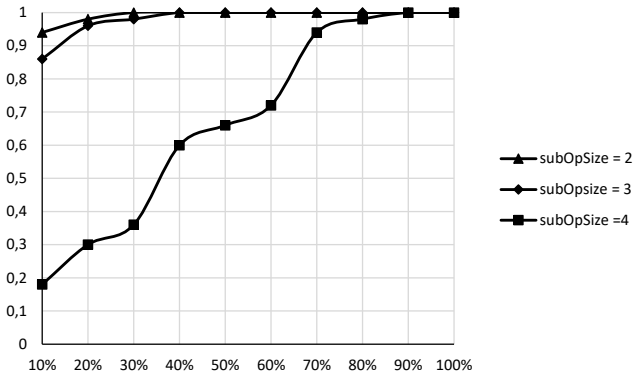

Figure 12. Solution probability

6.2.2. Quality of the negotiated solution

After considering the representatives, the quality of the negotiated solution should also be evaluated. For this purpose, a comparison is first made as to whether the negotiated solution is advantageous from a cost point of view compared to the non-negotiated combination of the optimal partial solutions. Then the solutions after the negotiation are assessed.

Negotiated Solution

When combining the optimal solutions for the subproblems, it becomes clear that the lack of collaboration in problem solving leads to the achievement of individual goals, but the overall solution itself leaves room for optimization. Most

strikingly, synergy effects of simultaneous maintenance cannot be exploited.

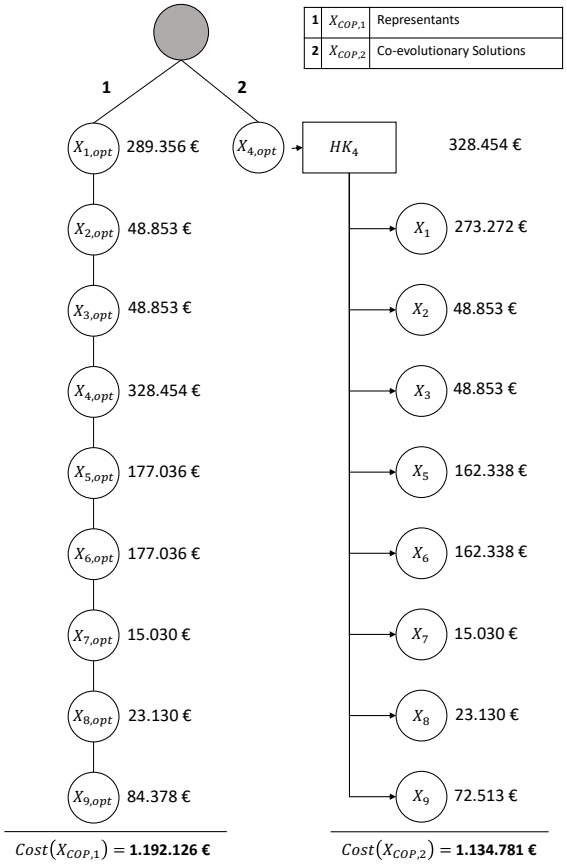

Figure 13. Cutout from the solution tree

The left branch in Figure 13 shows the result of finding solutions for the subproblems without boundaries (representants). A first iteration of the negotiation mechanism has shown that subproblem 4 is decisive for the search for a solution, i.e. the solution of the other subproblems must be based on subproblem 4. Thus the solution of subproblem 4 $X_{4,opt}$ was transferred to an auxiliary component HK according to chapter 4 and communicated to the *Subproblem-Solver* agents as a constraint. The results for each subproblem for the co-evolution is shown in Figure 13 (right branch). Especially by using synergy effects in maintenance, cheaper strategies for the subproblems could be discovered. The corresponding savings in comparison of both solutions are shown in the last column of Figure 13. The sense of the negotiation can thus be shown by a (fictitious) cost advantage of 57.345 Euro (nearly 5%) when applied. From the perspective of finding a compromise, one of the essential tasks of the negotiation, the following statements can be made: subproblem 2,3,4,7 and 8 did not change their position, whereas subproblems 1, 5, 6 and 9 were able to achieve a better goal (symmetric compromise).

7. CONCLUSION

In this paper, the optimization problem resulting from the coordination of the condition-based maintained components was presented. Based on this, a MAS was designed using the O-MaSE model, which heuristically solves the problem on the basis of co-evolutional distributed problem solution. Since there is no real validation scenario for the system, it was shown using criteria of solution quality and probability (Kirsch (1973)) that the MAS can deliver suitable solutions. The determination of the optimal problem size where heuristics still generate acceptable solutions and the comparison of non-negotiated and negotiated solution, which underlines the advantage of using negotiation for optimization in the context of the scenario presented in this paper, should be emphasized. The results of this paper can thus be seen as a first indication that distributed problem solving by using a MAS is a suitable approach to coordinate maintenance planning for complex systems. In order to strengthen this impression, the system has to be tested in real operation. However, an essential prerequisite for this is the availability of corresponding field data, which is currently not guaranteed (Missing Data). For this reason, in addition to testing the MAS, the focus should also be on generating corresponding data, since this forms the basis for further considerations.

APPENDIX

The following Table 1 is, as mentioned in section 6.1, the result of the data acquisition. The data is used as input for the optimization problem.

Table 1. Relevant vehicle data for the optimization problem

compID	Component	$Cost_M(compID)$	$Cost_M(compID)/\max(Cost_M)$	$DT(compID)$	$DT(compID)/\max(DT)$	$T(compID)$	$T(compID)/\max(T)$	sum
1	Flow Gear	144.000	1,00	120	1,00	30500	0,80	**1,62**
2	Engine (diesel)	98.800	0,69	42	0,35	47500	1,00	**1,26**
3	Axle Gear	81.800	0,57	60	0,50	30000	0,66	**1,01**
4	Axle Gear 2	81.800	0,57	60	0,50	30000	0,66	**1,01**
5	Axle Gear 3	81.800	0,57	60	0,50	30000	0,66	**1,01**
6	Compressor	45.533	0,32	7	0,06	40000	0,80	**0,86**
7	Axle Bearing 1	11.933	0,08	28	0,23	29500	0,66	**0,71**
8	Axle Bearing 2	11.933	0,08	28	0,23	29500	0,66	**0,71**
9	Axle Bearing 3	11.933	0,08	28	0,23	29500	0,66	**0,71**
10	Axle Bearing 4	11.933	0,08	28	0,23	29500	0,66	**0,71**
11	Axle Bearing 5	11.933	0,08	28	0,23	29500	0,66	**0,71**
12	Axle Bearing 6	11.933	0,08	28	0,23	29500	0,66	**0,71**
13	Axle 1	41.066	0,29	56	0,47	20000	0,33	**0,64**
14	Axle 2	41.066	0,29	56	0,47	20000	0,33	**0,64**
15	Axle 3	41.066	0,29	56	0,47	20000	0,33	**0,64**
16	Break Disk	26.333	0,18	28	0,23	8903	0,14	**0,33**
17	Elastic Coupling	11.366	0,08	11	0,09	12500	0,26	**0,29**
18	Wheel 1	12.833	0,09	28	0,23	2938	0,03	**0,25**
19	Wheel 2	12.833	0,09	28	0,23	2938	0,03	**0,25**
20	Wheel 3	12.833	0,09	28	0,23	2938	0,03	**0,25**
21	Wheel 4	12.833	0,09	28	0,23	2938	0,03	**0,25**
22	Wheel 5	12.833	0,09	28	0,23	2938	0,03	**0,25**
23	Wheel 6	12.833	0,09	28	0,23	2938	0,03	**0,25**
24	Cardan Shaft 1	4.500	0,03	2	0,01	5500	0,13	**0,14**
25	Cardan Shaft 2	5.400	0,04	3	0,03	5500	0,13	**0,14**
26	Brake Pads	10.733	0,07	7	0,06	2500	0,03	**0,10**

ACKNOWLEDGEMENT

The development of the MAS in this paper was funded by the Federal German Ministry of Economics and Energy within the funding program “Central Innovation SME” in the project “a3-Lok”, funding reference ZF4060716SS7.

REFERENCES

Association of German Engineers (2018): VDI 2653-Part 1: Multi-agent systems in industrial automation Fundamentals.

Association of German Engineers (2018): VDI 2653-Part 2: Multi-agent systems in industrial automation Development.

Association of German Engineers (2014): VDI 3663-Part 1: Simulation of systems in materials handling, logistics and production – Fundamentals.

Atamuradov, V.; Medjaher, K.; Dersin, P.; Lamoureux, B.; Zerhouni, N. (2017): Prognostics and Health Management for Maintenance Practitioners - Review, Implementation and Tools Evaluation. In: *International journal of prognostics and health management - Special Issue on Railways & Mass Transportation.*

Brenner, W.; Zarnekow, R.; Wittig, H. (1998): Intelligente Software-Agenten. Springer-Verlag.

DeLoach, S.; Garcia, J. (2014): O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. In: *IJAOSE (International Journal of Agent-Oriented Software Engineering)*. 3/2014. 244 – 280. DOI: 10.1504/IJAOSE.2010.036984

Fischer. J.; Kruschwitz, L.: Methodische Probleme bei der Evaluation heuristischer Lösungsverfahren. In: *Die Unternehmung*. 173 – 188. Nomos Verlagsgesellschaft.

Franzen, J.; Kuhlenkötter, B. (2018): Präskriptive Analyse im Kontext der Instandhaltung von Schienenfahrzeugen. In. *ETR*, 12/2018. Eurail Press.

Franzen, J.; Kuhlenkötter, B. (2019). Innovative Instandhaltung von Schienenfahrzeugen - Strategieoptimierung durch Einsatz des genetischen Algorithmus. In: *Internationales Verkehrswesen*, Journal 4/2019, Trialog Publishers.

Franzen, J.; Pinders, U.; Lingen, M.; Kuhlenkötter, B. (2019): Reduction of system-level lifecycle costs through movement-based operation adjustment for railway vehicles. In: *Proceedings of the 12th World Conference on Railway Research*, Tokyo, Japan.

Goodman, D.; Hofmeister, J.; Szidarovszky, F. (2019): Prognostics and health management. Wiley series in quality and reliability engineering.

Grigoriev, A.; van de Klundert, J.; Spieksma, F. (2006): Modeling and solving the periodic maintenance problem. In: European Journal of Operational Research (3) 172.

Kirsch, W.; Bamberger, I.; Gabele, E.; Klein, H. (1973): Betriebswirtschaftliche Logistik. Springer Fachmedien. DOI: 10.1007/978-3-322-89689-6.

Mansour, A. (2011): Solving the Periodic Maintenance Scheduling Problem via Genetic Algorithm to Balance Workforce Levels and Maintenance Cost. In: American J. of Engineering and Applied Sciences 4 (2).

Potter, M.; Jong, K. (2000): Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. In: Evolutionary Computation (8) 1.

Swanson, L.: Linking Maintenance Strategies to performance. In: *International Journal of Production Economics* (70). 237 – 244. Elsevier.

Sycara, K.; Decker, K.; Pannu, A.; Williamson, M.; Zeng, D.; Distributed Intelligent Agents. In: *IEEE Expert 1996*.

Weyns, D.; Michel, F. (2015): Agent Environments for Multi-agent Systems – A Research Roadmap. In: Agent Environments for Multi-Agent Systems IV. Lecture Notes in Computer Science, Vol. 9068. Springer, Cham. DOI: 10.1007/978-3-319-23850-0_1

Yang, Zhenyu; Tang, Ke; Yao, Xin (2008): Large scale evolutionary optimization using cooperative coevolution. In: Information Sciences (15) 178.

BIOGRAPHIES

Julian Franzen was born in Essen, Germany in 1990. He studied Mechanical Engineering with focus on automation (Ruhr-University Bochum, Germany, MSc 2015). He is currently PhD-Student at Ruhr-University Bochum. His research interest is the field of railway digitalisation, especially the realisation of proactive maintenance strategies.

Udo Pinders was born in 1966 and graduated in mechanical engineering in the department of design engineering with a focus on hydraulic and pneumatic machine technology (Dipl.-Ing.). In addition, he completed his studies as an industrial engineer (Dipl.-Wirt.-Ing.). He has been managing consulting and engineering offices for 25 years. Since 2002 he has been responsible for the technical management of the Westphalian locomotive factory Reuschling before he became managing director in 2020.

Bernd Kuhlenkötter Bernd Kuhlenkötter, born in 1971, studied mechanical engineering (TU Dortmund, Dipl.-Ing., 1997). After completing his studies, he was employed as a scientific assistant, senior engineer and head of department at the Department of Machine Elements, Design and Handling Technology. From 2005-2007 he represented the professorship Industrial Robotics and Handling Systems at the Institute for Robotics Research of the TU Dortmund. Afterwards, he was responsible for numerous national and international development projects as Development Manager at ABB Robotics. From 2009 to 2015, he was responsible for industrial work as Head of the Chair of Industrial Robotics and Production Automation (IRPA) at TU Dortmund. Since 2015, Prof. Kuhlenkötter heads the Chair of Production Systems (LPS) at the Ruhr University Bochum.