Self-Adaptive Ant Colony Optimisation Applied to Function Allocation in Vehicle Networks

Manuel Förster, Bettina Bickel
University of Erlangen-Nuremberg
Computer Science Department 2
Martensstrasse 3, 91058 Erlangen
gecco@manuelfoerster.net
Bettina.Bickel@methodpark.de

Bernd Hardung, Gabriella Kókai
University of Erlangen-Nuremberg
Computer Science Department 2
Martensstrasse 3, 91058 Erlangen
bernd@hardung.de
kokai@informatik.uni-erlangen.de

ABSTRACT

Modern vehicles possess an increasing number of software and hardware components that are integrated in electronic control units (*ECUs*). Finding an optimal allocation for all components is a multi-objective optimisation problem, since every valid allocation can be rated according to multiple objectives like costs, busload, weight, etc. Additionally, several constraints mainly regarding the availability of resources have to be considered. This paper introduces a new variant of the well-known ant colony optimisation, which has been applied to the real-world problem described above. Since it concerns a multi-objective optimisation problem, multiple ant colonies are employed. In the course of this work, pheromone updating strategies specialised on constraint handling are developed. To reduce the effort needed to adapt the algorithm to the optimisation problem by tuning strategic parameters, self-adaptive mechanisms are established for most of them. Besides the reduction of the effort, this step also improves the algorithm's convergence behaviour.

Categories and Subject Descriptors: I.2.8 [ARTIFICIAL INTELLIGENCE]: Problem Solving, Control Methods, and Search — Heuristic methods.

General Terms: Algorithms, Performance.

Keywords: Self-adaptation, ant colony optimization, automobile industry, multi-objective optimisation.

1. INTRODUCTION

Modern vehicles are provided with a great number of functions. For the implementation of these functions both software and hardware components are needed. During the development of the vehicle these components have to be assigned to *electronic control units* (*ECU*). Every valid assignment can be rated according to several quality criteria like costs, weight, busload, and supplier complexity. Finding an optimal assignment with respect to these criteria is a very complex task and due to an increasing number of functions almost impossible to handle manually. Therefore, it is a great facilitation of the development process, if an optimal mapping is computed automatically. As an important part of this automation, the effort of adapting the optimisation algorithm to specific problem instances by tuning strategy parameters ought to be reduced as far as possible. To achieve this, several proposals from literature were taken into consideration. After all, self-adaptation seemed to be the most promising option. This paper proposes a strategy by which an optimal assignment of all used functions can be found by the application of *Self-Adaptive Ant Colony Optimisation* (*SAACO*).

The article is organised as follows: Section 2 explains in detail the multi-objective optimisation problem, on which ACO is applied. Section 3 provides a brief overview of Ant Colony Optimisation in general. The application of ACO on the special problem at hand is described in Section 4. All necessary adaptions and extensions to this optimisation technique are introduced. Section 5 shows the results that were obtained by the application of the new ACO strategy. In section 6 the achieved results are discussed and a motivation for further research is given.

2. ALLOCATION OF FUNCTIONS IN VEHICLE NETWORKS

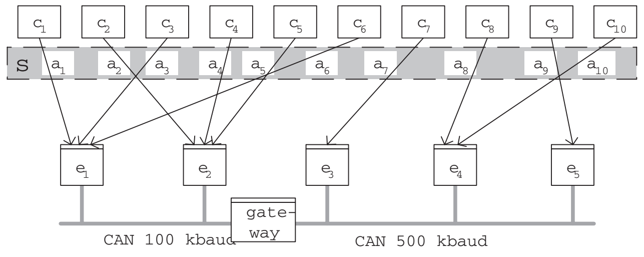
Every modern vehicle has a complex network of interacting functions, composed of so-called software and hardware components. These functions have to be assigned to ECUs. On an ECU, hardware components like network interfaces, power electronics, and pins are located, that allow the attachment of sensors and actuators[1]. Microcontrollers provide resources like ROM/RAM and timers, on which the assignment of software components is possible. The following two sections take a closer look on the assignment of these hardware and software components.

2.1 Allocation of Components

In the following we refer to *components* as an aggregation of software and hardware components. This aggregation allows to model the components, which depend on each other,

[1]We call all electronic components that are not inside the ECU sensors and actuators. Sensors normally provide information to the ECU (e. g. switches) and actuators normally execute commands from the ECU (e. g. motors or bulbs), see also [13].

as single components. It is a common case that actuators need power electronics directly connected on an ECU and that driver software must be on the same ECU as well, due to timing and/or safety requirements. In an early stage of the development process, these components have to be mapped to ECUs. For this purpose, several solutions are assessed and compared. More formal, every solution s can be defined as a set of q assignments $a = \{c \rightarrow e\}$, since any of the q components c has to be assigned to an ECU e. Therefore, every solution (see figure 1) is defined by $s = \{a_1, a_2, ..., a_q\}^{\mathrm{T}}$. Multiple objectives and constraints influence this decision, as will be described in the following section.



c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10}
S a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
e_1 e_2 e_3 e_4 e_5
gate-way
CAN 100 kbaud
CAN 500 kbaud

Figure 1: Schematic assembly of a solution

2.2 Objectives and Constraints

Several objectives have to be considered during the allocation of components. In this section a short overview of the ones mostly used is given:

- The *busload* depends on the allocation of components. To reduce the overall busload, functions with a high communication-rate can be located on the same networks or even on the same ECUs (see figure 1).
- Also the *electrical consumption* has to be considered. In the state *ignition off*, several networked functions are offered to the user, for example radio or hazard lights. Mapping components that have to be always active, like e.g. theft protection, on a preferably small number of ECUs, can reduce the electrical consumption.
- The next very important factor is *costs*. Costs can for example be decreased by reducing the length of cables in the wiring harness.
- Another objective to be considered is called *supplier complexity*. For every component, a set of suppliers, that is able to develop and produce it, is defined. For a given allocation and a given ECU, the supplier complexity is the minimum number of suppliers involved in the development of the ECU. The problem of calculating this minimum number can be transformed into a so-called *set covering problem*. Algorithms for solving it are shown in [2]. As a global value for the objective supplier complexity, we calculated the average of all ECUs.

Furthermore, many *pure constraints*[2] are involved in the search for optimal mappings, for example the consumption of memory, pins, timer, and space of the circuit on the ECU.

[2]Pure constraints are constraints that are not objectives at the same time. They are not necessary hard constraints.

In addition, constraints that are objectives also have to be considered. An example is busload, which is limited. Often, the boundary is much lower than 100%, since free capacity is reserved for additional functions that are introduced in a later phase of the product life cycle of the new car. Still busload is an objective, even if below the boundary, since a lower busload gives more degrees of freedom for the development of new functions in the future.

Not every new vehicle series demands a total redevelopment of the whole system architecture. The addition of some new functions is already a very demanding task. Considering ten new functions for a new vehicle series and five possible locations for each, already $5^{10} = 9765625$ possibilities for the allocation exist, although constraints reduce this number significantly. As can be seen later, under certain circumstances, only 0.002% of the possible combinations are valid at all.

3. ANT COLONY OPTIMISATION

Ant Colony Optimisation is a modern optimisation technique based on the principle that real ants are able to find the shortest paths between their nest and a food source [11]. This works on the basis of pheromones, some kind of biochemical scent, which is left behind by the ants. Other ants are attracted by these pheromones and always walk in the direction with the highest pheromone concentration. This natural behaviour was first adopted as an optimisation technique by Dorigo, Colorni, and Maniezzo [3, 7]. Their *Ant System* provides the foundation of the optimisation method presented here.

3.1 General Approach

Virtual ants communicate through pheromones, just as real ants. The typical task of virtual ants is finding the shortest path in a given graph. To each edge of the graph, a certain initial pheromone value is assigned representing the preference for choosing this edge. One single ant builds a complete path by choosing one edge after another according to the rule

$$p_{ij} = \frac{{\tau_{ij}}^{\alpha}\ {\eta_{ij}}^{\beta}}{\sum_{j \in N} {\tau_{ij}}^{\alpha}\ {\eta_{ij}}^{\beta}}\ . \qquad (1)$$

Here, τ_{ij} refers to the pheromone value, η_{ij} to the so-called heuristic information, which is specific for the particular problem, and N to the neighbourhood of node i. This equation describes the probability for choosing edge ij when starting from node i. The higher the values for pheromone and heuristic information the higher is the probability for choosing this edge. The weighting-parameters α and β adjust the influence of pheromone and heuristic information.

After having built a complete path, the pheromone values are updated. First, all values are decreased by a certain percentage subject to the *evaporation* of natural pheromones. This mechanism prevents the ants from settling on a certain path too fast. After evaporation has been taken into account, the ants increase the pheromones according to specific rules. Either all ants update the edges of their paths with an amount according to the length of their path or only the ant(s) having found the shortest path(s) are allowed to refresh the pheromone values. The intention there is that the edges of good paths with lower lengths get higher pheromone values so that they will be chosen again with a high probability by following ants.

The new pheromone value at the time $t + 1$ is determined with

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t, t+1), \qquad (2)$$

where ρ is a measure for the persistence of the pheromones. The increment $\Delta\tau_{ij}(t, t+1)$ is either a constant or depends on the length of the generated tour.

When the pheromone updating has taken place, all ants of the current generation die and new ants are created, that will again look for the shortest paths but now referring to new pheromone values. The idea of ACO lies in passing one generation's information to the next solely by means of the pheromone values. After generating many paths, the solutions that were found ideally converge to the optimum of the given problem.

So far, there has only been a global pheromone-update after all paths have been computed. But there is also the possibility to use a local pheromone-update each time an ant moves from one node to another [6]. Thereby the pheromone value of an edge is decreased when it has been chosen by an ant. The idea of this strategy is similar to the evaporation: ants of the following generations should not be affected too strongly by paths that were already discovered. Instead, they should be motivated to enter new, unexplored regions of the search space.

3.2 ACO for Multi-Objectives

To be able to address multi-objective optimisation problems several extensions to ACO have been developed (see, for example, [10], [15] and [14]). The most important one for this work uses multiple ant colonies [14], one for each objective. All colonies possess their own pheromone information, independent from each other. While the ants search their paths in the graph, they only use the pheromone information of their own colony. For achieving an indirect communication between the colonies, all found solutions are collected in a global set. Starting from this superior set, the current Pareto-front is determined. The Pareto-front covers all solutions that are not dominated by other solutions [5]. Since a multi-objective optimisation problem does not have only one single optimal solution, the Pareto-front is taken as the set of optimal solutions. The current Pareto-front is then used for updating the pheromones of the single colonies. Here the approach "*update-by-origin*" is applied, which means that every solution in the Pareto-front updates the pheromone values of its own colony.

3.3 Dynamic Parameter Adaptation

Conventional heuristic optimisation algorithms are sensitive to the settings of their specific strategy parameters. The algorithms' convergence behaviour is mainly determined by the setting of these parameters. The main purpose of parameter tuning for multi-objective optimisation is the gain of a trade-off between the algorithm's convergence rate and the diversity of the solutions it ends up with. Usually, multi-objective optimisation requires to have a choice between a variety of solutions. However, a high convergence rate usually has a negative impact on the variety of solutions, and vice versa. Heuristic algorithms that aim at a high convergence rate tend to end up in local optima quite early in the optimisation process, which is obviously not intended. Therefore, tuning the parameters can be considered as a very important part of applying an optimisation algorithm to a problem instance. Unfortunately, the optimisation of the parameters itself can be as complex as the original optimisation problem. Changing parameter values can cause unintended side-effects to the optimisation process. Usually, experienced users set those parameters in time-consuming trial-and-error sessions. The main goal of dynamic parameter optimisation for the algorithm described in this paper was reducing the effort needed for applying the algorithm to the optimisation problem described above. According to Eiben [1], the following different methods of parameter adaptation are available:

- Deterministic parameter control: Parameter values are changed during the optimisation process, according to predetermined rules.

- Adaptive parameter control: The way in which the parameter values are changed depends on feedback from the optimisation process, the adaptation takes place outside of the optimisation algorithm.

- Self-adaptive parameter control: The optimisation of the parameter values happens inside the optimisation algorithm, the same mechanisms that are used to optimise the original problem are applied to the problem of optimising the parameter values.

The main advantage of the adaptive and self-adaptive versions compared to the deterministic one is the ability to dynamically react to changes in the problem instance's requirements to the parameter values. After all, self-adaptive parameter control seemed the most promising option, as it enables to reduce the effort of parameter tuning, as well as it makes the algorithm more robust due to the ability to react to changes in the optimisation process. Literature provides sparse information on parameter tuning for ant colony optimisation. Pilat and White [16] suggest a hybrid implementation, combining a genetic algorithm with ant colony optimisation, where the genetic algorithm optimises the parameter values of the underlying ant colony optimisation. The main disadvantage of this option would be the additional burden that the genetic algorithm generates, so this method was not taken into consideration. Gaertner et al. [9] propose a comparable hybrid of genetic algorithm and ant colony optimisation, which also was rejected for the same reason as above. Randall [17] presents a promising proposal regarding self-adaptive ant colony optimisation. However, it has to be adapted to handle multi-objective optimisation problems properly. Mainly, the self-adaptive extension described in this paper is based on the concepts described in [17]. For the enhancements of *SAACO* regarding the parameter optimization, see section 4.4.

4. ACO FOR ALLOCATION OF FUNCTIONS IN VEHICLE NETWORKS

The Ant Colony Optimisation Technique was applied on the problem of allocating functions in vehicle networks. The following sections describe, which adaptions and extensions had to be made in order to optimise multiple objectives and handle constraints.

	c_1	c_2	c_3
e_1	τ_{c_1,e_1}	τ_{c_2,e_1}	τ_{c_3,e_1}
e_2	τ_{c_1,e_2}	τ_{c_2,e_2}	τ_{c_3,e_2}
e_3	τ_{c_1,e_3}	τ_{c_2,e_3}	τ_{c_3,e_3}
e_4	τ_{c_1,e_4}	τ_{c_2,e_4}	τ_{c_3,e_4}

	c_1	c_2	c_3
c_1	γ_{c_1,c_1}	γ_{c_2,c_1}	γ_{c_3,c_1}
c_2	γ_{c_1,c_2}	γ_{c_2,c_2}	γ_{c_3,c_2}
c_3	γ_{c_1,c_2}	γ_{c_2,c_3}	γ_{c_3,c_3}

Figure 2: Example for pheromone matrix (left) and correlation matrix (right) for three components and four ECUs

4.1 Correlation Matrix

The first and most important step is finding a suitable representation for the pheromone information. A straightforward approach is using a two-dimensional pheromone matrix with columns representing components and rows representing ECUs. The values in the matrix show the desirability for mapping one component to a special ECU. The drawback of this design is that dependencies between components cannot be displayed. The two-dimensional matrix can only store the preferences for mapping single components onto certain ECUs, whereas relations between components cannot be taken into account. For this reason an additional correlation matrix is introduced. The correlation matrix is also a two-dimensional matrix, but with columns and rows representing components. The values indicate the desirability of mapping two components together on one ECU.

Figure 2 shows an example of the pheromone and the correlation matrices for three components and four ECUs.

The probability $p_{c,e}$ for chosing a certain ECU e as allocation target for a component c, can be determined by

$$p_{c,e} = \frac{\tau_{c,e} \prod\limits_{\forall\gamma\in\bar{\gamma}'} \gamma}{\sum\limits_{\forall c} \tau_{c,e} \prod\limits_{\forall\gamma\in\bar{\gamma}'} \gamma}, \tag{3}$$

where $\bar{\gamma}'$ contains all γ_{c_1,c_2} where $c_1 = c$ and c_2 is already allocated on e.

4.2 Applying Multiple Colonies

The function allocation problem is a multi-objective optimisation problem. As shown in section 3.2, there are methods for handling multiple objectives in ACO. The algorithm shown here is an extended version of the work of Iredi et. al. [14]. Several colonies are created, but not necessarily one for each objective. The number of colonies is configurable. Main advantage of applying multiple colonies is that the colonies can work on different areas of the search space, which increases the possibility of finding new and/or better solutions.

Figure 3 shows the hierarchy of different groups of ants. The different groups of ants form a hierarchy. The lowest rank is taken by the *ant* itself. Its task is to generate a complete solution. Using the pheromone and correlation matrices of its colony it allocates all components to a specific ECU. The *ant colony* is the superior group of ants. It controls the individual ants and administrates the pheromone and correlation values. At the highest level, the *ant population* is found. It collects the generated solutions, evaluates and compares them and controls the update of the matrices. The ant population stores the best solutions in a *global solution set*. The definition of what the *best* solutions are, is given below.

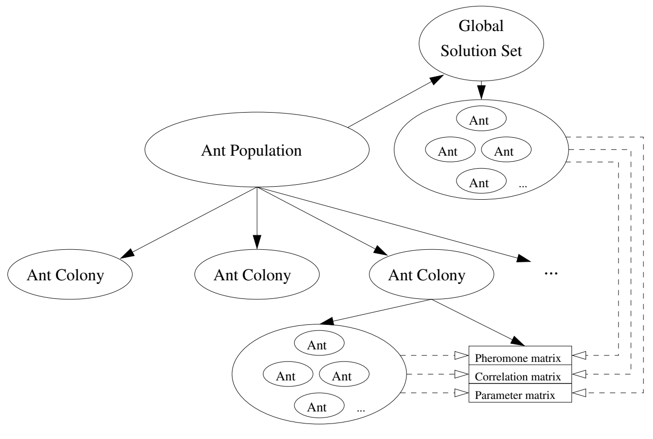


Figure 3: Hierarchy of self-adaptive ACO

4.3 Strategies for Pheromone Updating and Constraint-Handling

It has already been described above that the pheromone values are updated everytime a generation of ants has computed their solutions. The first step that has to be performed is called evaporation. The evaporation reduces the pheromone and correlation values by a certain configurable percentage.

This approach realises the strategy that only the best ants are allowed to map their information onto the pheromone values. It has already been said that a multi-objective problem does not have one single best solution but rather a set of optimal solutions which can be referred to as Pareto-front. Since the problem at hand also has several constraints that have to be fulfilled, it does not suffice to use the Pareto-solutions for updating. It turned out to provide good results, if solutions were taken into account, that do not violate any constraints.

Besides, the best solutions of all generations so far are collected in a assured that good information cannot get lost again. In fact, there are three solution sets which are used for updating: the Pareto-solutions of the current generation, the feasible solutions of the current generation (i.e. those that do not violate any constraint) and the solutions contained in the global set. The amount by which the pheromones are increased can be configured by the user.

This update process can be rewritten as

$$\begin{aligned}\tau_{c,e}(t+1) = {} & (1-\mathfrak{p}_{\text{PER}})\tau_{c,e}(t) \\ & + \Delta\tau_{c,e}(t,t+1)\end{aligned} \tag{4}$$

and

$$\begin{aligned}\gamma_{c_1,c_2}(t+1) = {} & (1-\mathfrak{p}_{\text{CER}})\gamma_{c_1,c_2}(t) \\ & + \Delta\gamma_{c_1,c_2}(t,t+1),\end{aligned} \tag{5}$$

where $\tau_{c,e}(t)$ denotes the "old" pheromone value at the time t. The parameter $\mathfrak{p}_{\text{PER}}$ describes the amount of evaporation and $\Delta\tau_{c,e}(t,t+1)$ refers to the increment of the pheromone value. The correlation values are updated analogously (see equation 5).

The values $\Delta\tau_{c,e}$ and $\Delta\gamma_{c_1,c_2}$ in equation 4 and equation 5 are determined at any time by

$$\begin{aligned}\Delta\tau_{c,e} &= \mathfrak{p}_{\mathrm{UGS}} p(c \to e, \bar{s}_g) \\ &+ \mathfrak{p}_{\mathrm{ULF}} p(c \to e, \bar{s}_f) \\ &+ \mathfrak{p}_{\mathrm{ULN}} p(c \to e, \bar{s}_n) \qquad (6)\end{aligned}$$

and

$$\begin{aligned}\Delta\gamma_{c_1,c_2} &= \mathfrak{p}_{\mathrm{UGS}} p(c_1, c_2, \bar{s}_g) \\ &+ \mathfrak{p}_{\mathrm{ULF}} p(c_1, c_2, \bar{s}_f) \\ &+ \mathfrak{p}_{\mathrm{ULN}} p(c_1, c_2, \bar{s}_n), \qquad (7)\end{aligned}$$

where $p(c \to e, \bar{s})$ denotes the probability that c is allocated to e in a solution set $\bar{s}$ and $p(c_1, c_2, \bar{s})$ the probability that c_1 and c_2 are allocated to the same ECU. The parameters $\mathfrak{p}_{\mathrm{UGS}}$, $\mathfrak{p}_{\mathrm{ULF}}$, $\mathfrak{p}_{\mathrm{ULN}}$ are adjustable by the user and determine the influence of the three solution sets (g - global, f - feasible, n - non-dominated). The same holds for the parameters in equation 7 concerning the correlation.

The last important decision is the usage of the *update-by-step* algorithm. The update-by-step algorithm decreases the value in the matrices every time an ant has mapped a component to an ECU. This way, the following ants will less likely follow the same path and the probability for exploring new regions of the search space is higher.

The global solution set also has to be updated in every generation. For this purpose, all the solutions of the current generation are added to the "old" global set. Then the size of the set is reduced again with a special truncation operator. This operator is based on a dominance relation developed especially for constraint-handling (originally from [4], modified in [12]). If there are only invalid solutions in the "new" global set, the truncation operator removes the solutions that have the strongest constraint violations. If there are valid solutions, but their number is smaller than the configured size of the global set, the set is filled with "good" invalid solutions. If there are more valid solutions than needed, the best solutions according to the second criterion are kept. With this strategy, the solutions that do not violate any constraints have a great influence on the update process of the matrices and thus the search of the ants is forced towards valid regions of the search space.

4.4 Self-Adaptive Parameters

The implementation of ant colony optimisation described above relies on the following ten different parameters:

- The *exploitation random parameter* ($\mathfrak{p}_{\mathrm{ERP}}$), which denotes the probability that the next step in constructing the tour is made without taking the pheromone values into account
- The parameter *colonies per objective* ($\mathfrak{p}_{\mathrm{NCPO}}$)
- The *subpopulation factor* ($\mathfrak{p}_{\mathrm{SPF}}$), which determines the number of ants per colony
- The *pheromone evaporation rate* ($\mathfrak{p}_{\mathrm{PER}}$), that means the amount by which the values in the pheromone matrix are decreased in the evaporation phase
- The *correlation evaporation rate* ($\mathfrak{p}_{\mathrm{CER}}$), which means the same for the correlation matrix
- The *population size factor* ($\mathfrak{p}_{\mathrm{PSF}}$), that determines the size of the global solution set

Parameter	*lb*	*ub*
Update local	0.0005	0.01
Update pareto	0.0005	0.01
Update global	0.0005	0.01
Exploitation random	0.05	0.15
Update by step	0.9	1.0
Pheromone evaporation	0.03	0.07
Correlation evaporation	0.08	0.16

Table 1: Parameter ranges

- The *local*, *global* and *pareto update factor* ($\mathfrak{p}_{\mathrm{ULF}}$, $\mathfrak{p}_{\mathrm{UGS}}$, $\mathfrak{p}_{\mathrm{ULN}}$), which denote the amount by which the pheromone and correlation values are increased when the update of the matrices is performed
- The *update by step* ($\mathfrak{p}_{\mathrm{FUP}}$) factor, by which the pheromone values are decreased right after constructing a tour, as described above

With the self-adaptive enhancements to the original ACO described in this work, seven of the parameters described above no longer need to be tuned by the user. It seemed not very promising to tune the parameters *subpopulation factor*, *colonies per objective*, and *population size factor* dynamically, so they still need configuration. The other parameters now are completely self-adaptive, and initialized by the mean value of their ranges. Therefore, there is no longer the need to consider the correct setting of those values when applying the optimisation algorithm. In order to apply the algorithms optimisation mechanisms to its own parameter values, an appropriate representation of the parameter information is needed. The pheromone matrix and the correlation matrix shown above are used to store the information on how function allocation should happen. Therefore, the values in each matrix represent probabilities for mapping components to ECUs. Using a comparable mechanism for representing parameter values, an additional matrix, the *parameter matrix*, is introduced. It provides information for mapping possible parameter values to the specific parameters. Figure 2 shows an example for a parameter matrix. As proposed in [17], ranges for the parameters are introduced, and, determined by a granularity value G, those ranges are split into several intervals. The lines of the matrix (s_g) now show the possible steps for the parameters which are represented by the columns (p_i). For the granularity G, 20 was found to be a suitable value.

The parameter ranges used for specifying the intervals that are represented in the parameter matrix were found empirically, starting with the boundaries of the values used for the *Robust Design* method [18]. Bit by bit, the boundaries of the ranges were expanded, until a set of useful intervals was found (see figure 1, where *lb* means the lower boundary, and *ub* the upper boundary of the parameter range).

Before the ants build tours through the pheromone and correlation matrices, they also build a tour through the parameter matrix. The same strategies for updating the pheromone in the first two matrices are used for updating the parameter matrix. Therefore, only little additional burden is added to the optimisation process.

The parameter values used by the optimisation algorithm now are computed by equation 8, where p_i denotes the parameter, $v(p_i)$ the value of the parameter, t_i the step in the

	p_1	p_2	p_3	...	p_i
s_1	π_{p_1,s_1}	π_{p_2,s_1}	π_{p_3,s_1}	...	π_{p_i,s_1}
s_2	π_{p_1,s_2}	π_{p_2,s_2}	π_{p_3,s_2}	...	π_{p_i,s_2}
s_3	π_{p_1,s_3}	π_{p_2,s_3}	π_{p_3,s_3}	...	π_{p_i,s_3}
...	...	...	...	...	...
s_g	π_{p_1,s_g}	π_{p_2,s_g}	π_{p_3,s_g}	...	π_{p_i,s_g}

Table 2: Example for parameter matrix

parameter matrix, G the granularity, $l(p_i)$ the lower and $u(p_i)$ the upper boundary of the parameter's range.

$$v(p_i) = l(p_i) + \frac{t_i}{G}(u(p_i - l(p_i))) \quad (8)$$

To completely avoid the effort for configuring parameter values, the parameters are initialized by the midpoint of their ranges.

5. RESULTS

The optimisation technique introduced here was applied to the following real-world example: a total of 23 components modeling the functions *keyless entry, central door locking, direction indication* and *exterior light* have to be allocated to 5 ECUs optimising the objectives costs, busload, electrical energy consumption and supplier complexity. Additionally, constraints regarding the memory and space resources must not be violated. In this example only 0.002 % of all possible solutions are valid.

As shown in the previous section, the Ant Colony Optimisation technique uses several parameters, that have to be adjusted correctly to achieve optimal results. For evaluating the self-adaptive enhancements that were described in section 4.4, the results were compared to static parameter settings determined by Taguchi's *Robust Design* method [18].

To point out the advantages of applying self-adaptive ACO to the optimisation problem described above, the results obtained during the optimisation were compared with those of a variation of the *SPEA2*, a well-known multi-objective evolutionary algorithm [19] (as well as an extended, self-adaptive implementation variant of *SPEA2*). The static implementations of the algorithms are introduced in [12], for the self-adaptive extensions to the evolutionary algorithm and the ant colony optimization, see [8].

Each of the different algorithms was run in 20 repetitions on the problem described above. The plots in this section are based on the average of those 20 runs. The average execution time for 20 repetitions of any of the algorithms described here is about four hours on a Pentium 4 at 1,8 GHz with 768 Megabyte of RAM. Adding self-adaptation does not influence execution times considerably. As a termination condition the number of objective evaluations was employed (set to a maximum of 20.000 evaluations).

The results are rated according to the so-called *hypervolume*, which is a measure for evaluating the quality of Pareto-fronts introduced in [20]. The lower the hypervolume values, the better the solution set. To compare the different optimisation algorithms regarding the diversity of the produced solutions, *boxplots* (as you can see in figure 6 and 8) are used. Here, the box contains 50% of the results (where result means the last hypervolume value produced by each of the 20 repetitions), 25% reside above the upper horizontal boundary, and the other 25% below the lower horizontal line. The vertical lines represent the peak values. So, the flatter the box is, the lower is the diversity of the generated hypervolumes. The horizontal line inside of the box denotes the median of the corresponding results.

Figure 4 shows the comparison of the variants of both algorithms with static parameters.

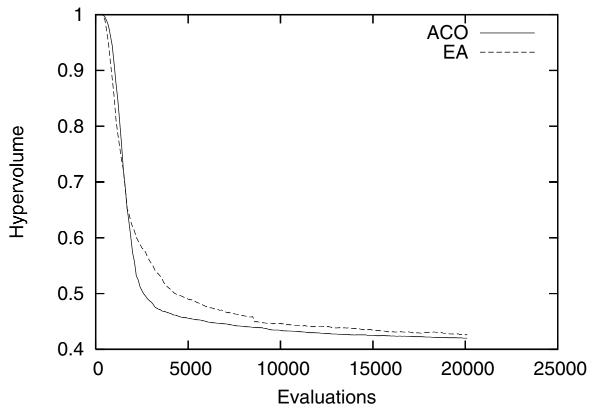


Figure 4: Comparison of the hypervolume values obtained with ACO and SPEA2

In figure 5, *SAACO* is compared to its static predecessor. Obviously, the self-adaptive extension results in a better convergence rate from the very beginning of the optimisation process.

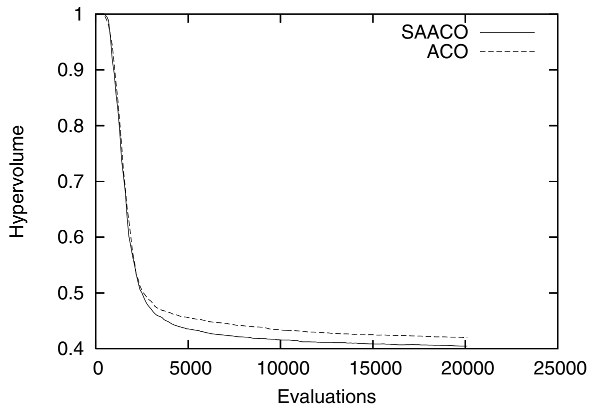


Figure 5: Comparison of the hypervolume values obtained with ACO and SAACO

Additionally, figure 6 shows a comparison of the distribution of the self-adaptive ACO and the static one. Not only the convergence rate, but also the distribution of the different results is improved by introducing self-adaptation.

To compare *SAACO*, the evolutionary algorithm based on *SPEA2* was extended with self-adaptive parameters as well. The results of the self-adaptive variant compared to the static one is shown in figure 7. At the beginning of the optimisation process, the self-adaptive algorithm performs worse than the static one. This is due to the addi-

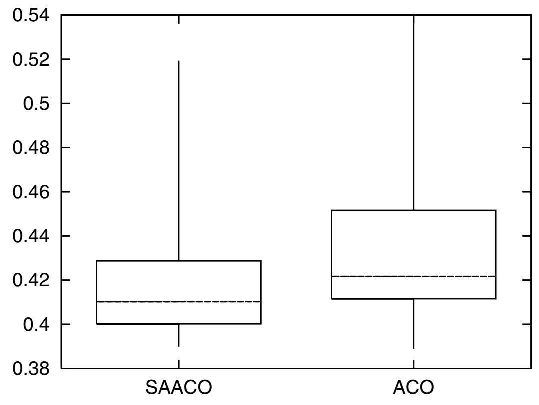


Figure 6: Comparison of the distribution of hypervolume values obtained with ACO and SAACO

tional effort it takes the algorithm to find optimal parameter values. One of its parameters, the mutationrate, has a comparatively high parameter range, which levels off after few thousand evaluations. Here also, the distribution of the

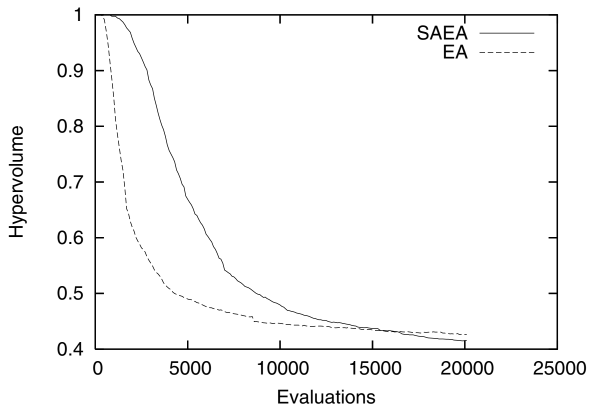


Figure 7: Comparison of the hypervolume values obtained with EA and SAEA

different results of each optimisation process is compared for the static and the self-adaptive algorithm, see figure 8. Finally, figure 9 shows a comparison of SAACO and SAEA. For the corresponding boxplot, see figure 10. As can be seen, SAACO performs considerably better than SAEA throughout the whole optimisation process.

To provide a comprehensive overview of the advantages of the self-adaptive extensions, table 3 shows the absolute hypervolume values produced by the different optimisation algorithms, and table 4 shows the proportion of the hypervolume values compared to each other. As can be seen from the values in both tables, all the algorithms perform better than the static EA, as well as they are all outperformed by the SAACO.

6. CONCLUSION

In this paper a new variant of the Ant Colony Optimisation technique is introduced. Particular work is done for adapting this method to multi-objective optimisation problems with constraints. The optimisation technique developed here is applied to a complex real-world problem, the allocation of software and hardware components to ECUs in vehicles. The results show that the ACO technique is qualitatively better than other optimisation techniques like the tested variant of SPEA2. Additionally, self-adaptive parameters for both optimisation algorithms have been introduced, which results in better performance, considering the convergence rate as well as the diversity of the different results. So far, heuristic information has not been used in this ACO variant. By defining reasonable heuristic information the technique could be adapted to the problem at hand even better. In addition, a combination with local search techniques for improving the results even further might be considered.

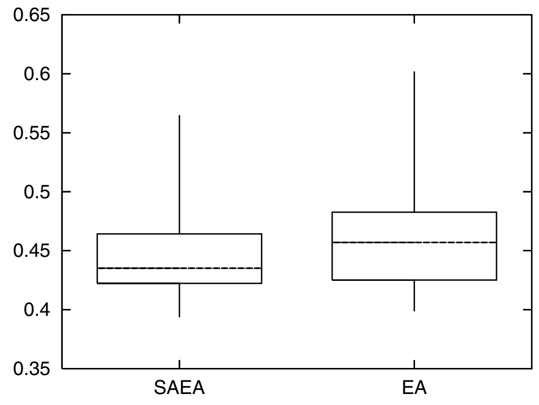


Figure 8: Comparison of the distribution of hypervolume values obtained with EA and SAEA

Algorithm	Average hypervolume value
EA	0.42612
ACO	0.41992
SAEA	0.41532
SAACO	0.40469

Table 3: Hypervolume values produced by the different optimisation algorithms

	EA	*ACO*	*SAEA*	*SAACO*
EA	1.0	1.015	1.026	1.053
ACO	0.985	1.0	1.011	1.038
SAEA	0.975	0.989	1.0	1.026
SAACO	0.950	0.964	0.974	1.0

Table 4: Improvements of self-adaptive extensions compared to original algorithm variants

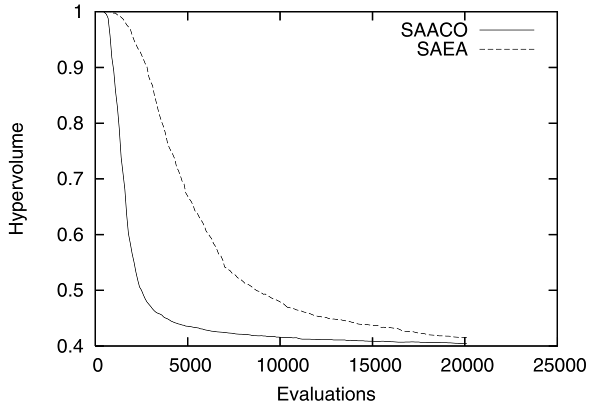


Figure 9: Comparison of the hypervolume values obtained with SAACO and SAEA

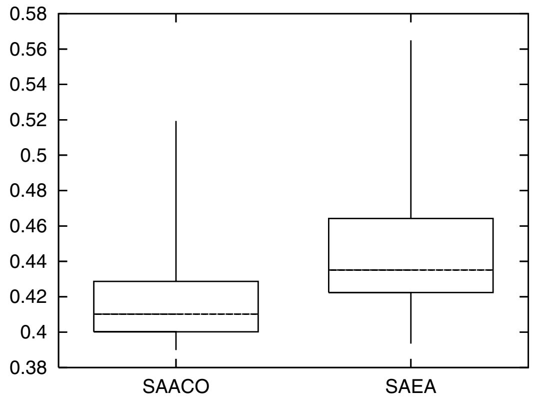


Figure 10: Comparison of the distribution of hypervolume values obtained with SAACO and SAEA

7. REFERENCES

[1] A. E. Eiben, R. Hinterding and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Trans. on Evolutionary Computation*, 3(2):124–141, 1999.

[2] A. Caprara, M. Fischetti, and P. Toth. Algorithms for the Set Covering Problem. Technical Report No. OR-98-3. DEIS-Operations Research Group., 1998.

[3] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. *Proceedings of the European Conference on Artificial Life, Paris, France, Elsevier Publishing*, pages 134 – 142, 1991.

[4] K. Deb. An efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, (186):311 – 338, 2000.

[5] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, Ltd., Chichester, UK, 2001.

[6] M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 1997.

[7] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(1):1 – 13, 1996.

[8] M. Förster. *Dynamische Parameteroptimierung für evolutionäre Verfahren zur Anwendung auf ein Mehrkriterienoptimierungsproblem mit Nebenbedingungen, Diploma thesis.* Friedrich-Alexander University Erlangen-Nuremberg, Computer Science Department 2, 2006.

[9] D. Gaertner and K. Clark. On optimal parameters for ant colony optimization. In H. Arabnia and R. Joshua, editors, *Proceedings of the International Conference on Artificial Intelligence 2005*, volume 1, pages 83–89. CSREA, 2005.

[10] L. M. Gambardella, E. Taillard, and G. Agazzi. Macs-vrptw: a multiple ant colony system for vehicle routing problems with time windows. pages 63–76, 1999.

[11] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels. Self-Organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, (76):579 – 581, 1989.

[12] B. Hardung. *Optimisation of the Allocation of Functions in Vehicle Networks, Dissertation.* Friedrich-Alexander University Erlangen-Nuremberg, Computer Science Department 2, 2006.

[13] B. Hardung, T. Kölzow, and A. Krüger. Reuse of software in distributed embedded automotive systems. In *Proceedings of the 4th ACM International Conference on Embedded Software, EMSOFT*, pages 203–210, Pisa, Italy, Sept. 2004.

[14] S. Iredi, D. Merkle, and M. Middendorf. Bi-Criterion Optimization with Multi Colony Ant Algorithms. *Proceedings of the First International Conference on Evolutionary Multicriterion Optimization*, pages 359–372, 2001.

[15] C. E. Mariano and E. Morales. A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks. Technical Report HC-9904, 1999.

[16] M. L. Pilat and T. White. Using Genetic Algorithms to optimize ACS-TSP, 2002.

[17] M. Randall. Near parameter free ant colony optimisation. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *ANTS Workshop*, volume 3172 of *Lecture Notes in Computer Science*, pages 374–381. Springer, 2004.

[18] G. Taguchi. *Introduction to Quality Engineering: Designing Quality into Products and Processes.* Asian Productivity Organization, Japan, 1986.

[19] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Eidgenössische Technische Hochschule Zürich (ETH), Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[20] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov. 1999.