This is the author's version of the work. The definitive work was published in *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS 2008)*, pp. 74-81, 2008. The work is supported in part by the German Science Foundation (DFG), SFB 694.

Multi-Objective Routing and Topology Optimization in Networked Embedded Systems

Michael Glaß, Martin Lukasiewycz, Rolf Wanka, Christian Haubelt, and Jürgen Teich
Hardware/Software Co-Design, Department of Computer Science
University of Erlangen-Nuremberg, Germany
Email: {glass, martin.lukasiewycz, rwanka, haubelt, teich}@cs.fau.de

***Abstract*— Modern networked embedded system design has to cope with multiple design objectives. One major challenge is the determination of optimal routings with respect to these objectives. Existing automatic optimization approaches carry out a two step optimization: First, they perform a multi-objective topology optimization of the networked embedded system. Then, a multi-objective routing optimization for a subset of Pareto-optimal solutions obtained from the first step is performed. In general, this may exclude several globally optimal solutions from the optimization process. To overcome this drawback, a unified approach based on Multi-Objective Evolutionary Algorithms is presented that ensures a combined optimization of the topology and routing. Since the system topology is varied within the optimization, the main contribution of this paper contribution is a novel routing technique that always samples feasible paths using a topology independent genetic encoding. This encoding preserves optimized routing information when changing the underlying topology. An experimental evaluation shows the effectiveness of the presented approach.**

I. Introduction

Today's embedded systems can be found in consumer products, prototyping systems or applications from automotive and avionics industries. Especially in automotive and avionics systems, there are many Electronic Control Units (ECUs) that are connected via different types of shared buses. These systems are commonly known as *networked embedded systems*. Design alternatives are not only given due to computational resources anymore, but include communication over several communication standards like LIN [15], CAN [6], FlexRay [11] or TTP [20]. This highly increases the design complexity of the networked embedded system.

Typically, the networked embedded system is a distributed system concurrently executing communicating processes that are statically bound onto computational resources in the network. A system-level designer, hence, has to take several issues into account: Which computational and communication resources are required, where to execute tasks on the given computational node, and how to route data through the network such that no overload occurs on resources and the capacity of communication links is not exceeded. Of course, all these decisions should be taken into account by optimizing different and often competitive objectives, like minimization of monetary costs, power consumption or maximizing fault-tolerance while fulfilling different constraints.

In this paper, networks consisting of computational resources that are able to execute a certain amount of software load, and links with a certain capacity for the communication demand between the functions are considered. In contrast to previous approaches, where the optimization process was divided into a two step process, it is not distinguished between a so-called *topology optimization*, where the binding of tasks on computational resources is performed, and a *routing optimization*, where the focus is an optimal communication for a given architecture and process binding. Hence, a combination of these two optimization approaches into a unified optimization process using state of the art *Multi-Objective Evolutionary Algorithms* (MOEAs) [3] is presented. Thus, a true multi-objective optimization is performed that includes the allocation of resources, the binding of processes and a calculation of a routing to ensure a correct communication. By respecting multiple objectives, the introduced methodology determines a set of so-called *Pareto-optimal* solutions that allows for an unbiased decision making.

The proposed optimization approach is outlined in Algo. 1. The core functionality is performed by a three step decoding technique that a) determines an allocation α of resources, b) performs a binding β of tasks onto the allocated resources, and c) determines a feasible static routing γ for all communication demands. Thus, each decoded solution corresponds to a complete networked embedded system that can be evaluated with respect to all given objectives. In contrast to approaches based on several steps where only a subset of objectives is used to preselect to individuals for a further synthesis step, the presented one step approach avoids to miss global optima. This is obvious since the optimality of solutions from previous optimization steps does not guarantee an optimality of the solutions that can be synthesized based on the already found solutions. With the three step encoding technique, a common MOEA procedure including selection of the fittest individuals as well as crossover and mutation operations can be used to perform the optimization.

The challenge of this unified approach is the changing topology that the routing optimization has to cope with. Since allocation as well as binding may alter the topology significantly, the used routing optimization has to be able to deal with changing underlying topologies. For this purpose, a new multi-objective routing approach that is based on the sampling of feasible paths is presented. These paths are generated

Algorithm 1 The optimization process including the three-step decoding.

```
1:  procedure EA
2:      while generation < maxGenerations do
3:          for each individual ∈ population do
4:              individual.decodeAllocation()
5:              individual.decodeBinding()
6:              individual.decodeRouting()
7:              individual.evaluateObjectives()
8:          end for
9:          population = selectNextGeneration(population)
10:         population.mutate()
11:         population.crossover()
12:         generation++
13:     end while
14: end procedure
```

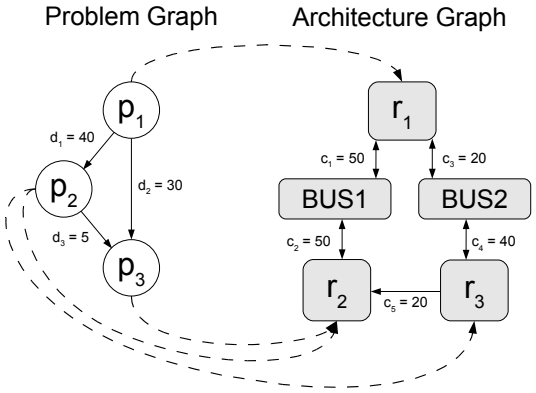

Fig. 1. Specification Graph: Edges in the problem graph are annotated with demands d_e. Edges in the architecture graph are annotated with capacities c_e.

via genetically encoded decision values and weighted routing hops. Furthermore, two algorithms are introduced that perform a fast and accurate weighting of hops based on matrices.

The remainder of the paper is outlined as follows: Section II discusses prior work. Section III introduces the used system model of the networked embedded system. Section IV gives a description of the problem targeted in this paper. Section V introduces the proposed path sampling algorithm while Sec. VI presents two weighting algorithms for the path sampling procedure. Experimental results are presented in Sec. VII while the paper is concluded with a summary in Sec. VIII.

II. Related Work

Several approaches exist that target a similar problem which is commonly referred to as *design space exploration* of embedded systems, e.g., [9], [10], and [16]. Unfortunately, these approaches are destined for SoC or MPSoC designs whereas no straightforward extensions exist for exploring the implementation alternatives of networked embedded systems as they occur in automotive, avionic, or wireless sensor networks.

In [19], a framework has been proposed that is able to satisfy the demand for an exploration of network embedded systems. The main drawback of this solution is the absence of a true multi-objective routing when exploring the design space. Hence, a heuristic is used that is based on routing over shortest paths from source to sink with the number of hops being the weight of a path. This heuristic makes sense since many objectives of networked embedded systems like the network flow, latency or fault-tolerance are often correlated with the number of hops to be taken. A comparison of the heuristic from [19] with the multi-objective routing approach proposed in this work is given in Sec. VII.

On the other hand, there are several multi-objective routing approaches that focus on so-called *Multi-Objective Routing Problems* (MORPs) [5]. In [2], [7], and [14], Evolutionary Algorithms are used to search for Pareto-optimal multi-cast routings for a given topology. Hence, all these approaches use a genetic encoding of the routing-paths for a fixed topology including resources and communication connections. As already mentioned, the approach proposed in this work has to deal with changing topologies and bindings from the first two decoding steps. Using the encodings proposed in [2], [7], [14], a change of the topology would imply a new encoding of the routing and therefore, a loss of all information for the routing optimization.

In [12], a framework called *UMARS* is presented that combines mapping of cores and routing of communication demands for *Network-on-Chip* architectures. Although this approach combines a mapping optimization with a routing optimization, the topology of the NoC is fixed. Therefore, the need for a topology independent routing is not given in this approach.

To the best of our knowledge, besides the approach in [19], there exists no framework for the topology optimization of networked embedded systems. Furthermore, there is no evolutionary driven multi-objective routing approach that can handle different underlying topologies without changing its genetic encoding and therefore losing information or requiring a huge amount of analysis to find reusable genetic information.

III. System Model

The used system *specification* strictly separates behavior and structure and is represented by a *specification graph* [4].

- The specification graph $G_\mathrm{s} = (G_\mathrm{p}, G_\mathrm{a}, E_\mathrm{m})$ consists of a *problem graph* G_p, an *architecture graph* G_a and *mapping edges* E_m.
- The *problem graph*[1] $G_\mathrm{p} = (V_\mathrm{p}, E_\mathrm{p})$ consists of processes $p \in V_\mathrm{p}$ and their data dependencies $e = (p_i, p_j) \in E_\mathrm{p} \subseteq V_\mathrm{p} \times V_\mathrm{p}$ and represents the *application* that is executed in the networked embedded system. The given data dependencies have to be implemented by routing the data from process p_i to p_j.
- The *architecture graph* $G_\mathrm{a} = (V_\mathrm{a}, E_\mathrm{a})$ consists of a set of available resources $r \in V_\mathrm{a}$ and their interconnections

[1] In [13], it has been shown how applications can be represented by a problem graph.

$e = (r_i, r_j) \in E_\mathrm{a} \subseteq V_\mathrm{a} \times V_\mathrm{a}$. This graph represents the *architecture* that is available to implement the networked embedded system.
- The mapping edges $m \in E_\mathrm{m} \subseteq V_\mathrm{p} \times V_\mathrm{a}$ relate processes of the problem graph G_p to resources of the architecture graph G_a and indicate a possible implementation of a process on the corresponding resource.

An example of a specification graph is shown in Fig. 1. White resources connected via directed edges represent processes with their data dependencies while gray resources represent the available resources connected via directed edges. The dashed edges in Fig. 1 depict the mapping edges.

With each data dependency $e \in E_\mathrm{p}$, a *demand value* $d_e \in \mathbb{R}_0^+$ is associated that represents the required bandwidth. On the other hand, the supported bandwidth is modeled by *capacities* $c_e \in \mathbb{R}_0^+$ associated with the interconnections $e \in E_\mathrm{a}$ in the architecture graph. Therefore, the routing possibilities of a data dependency $e \in E_\mathrm{p}$ are limited to connections $\widehat{e} \in E_\mathrm{a}$ that have enough bandwidth, hence fulfilling $d_e \leq c_{\widehat{e}}$. Furthermore, tasks, resources, and mapping edges can be attributed to model their properties for the evaluation of the objectives like monetary costs, network flow, latency, reliability, or power consumption. For a further explanation of the system model, cf. [19].

IV. Problem Description

In this section, the problem targeted in this paper is introduced. Afterwards, the need for a topology independent routing technique that results from the first decoding steps of allocation α and binding β is explained.

A. Design Space Exploration

The task of *design space exploration* is to find the set of optimal *feasible implementations* for a given *specification*. An implementation or *solution* of the problem is deduced from the specification and consists of three main parts:

- The *allocation* $\alpha \subseteq V_\mathrm{a}$ is the subset of available hardware resources that are actually used to implement the networked embedded system.
- The *binding* $\beta \subseteq E_\mathrm{m}$ determines on which allocated resource each process is executed. For each process from the problem graph exactly one mapping has to be in use.
- The *routing* γ contains a communication *path* through the network for each data dependency $e \in E_\mathrm{p}$ that, thus, allows a data transfer between the data dependent processes.

Due to the data dependencies, the demand values, and the capacities of the interconnections, an implementation can be infeasible. An implementation is called feasible if every process is mapped to an allocated resource and every data dependency imposed by the problem graph could by successfully established with a routing on the given resources with respect to the demand values and interconnection capacities.

Definition 1 (Design Space Exploration): The task of design space exploration can be formulated as the following multi–objective optimization problem:

$$\begin{aligned}&\text{minimize } f(\alpha, \beta, \gamma),\\ &\text{subject to:}\\ &\quad \alpha \text{ is a feasible allocation,}\\ &\quad \beta \text{ is a feasible binding,}\\ &\quad \gamma \text{ is a feasible routing}\end{aligned}$$

B. Routing

Since the focus of this work is on the routing, a formal introduction is given in the following. Per definition, a feasible routing γ contains a path γ^e for each data dependency $e = (p_s, p_t) \in E_\mathrm{p}$:

$$\gamma = \{\gamma^e \mid e \in E_\mathrm{p}\} \tag{1}$$

A *path* $\gamma^e \in \Gamma^e_{\alpha,\beta}$ is defined as a vector of resources V_a that use their point-to-point interconnection to pass the data. Here, $\Gamma^e_{\alpha,\beta}$ denotes the set of all feasible paths on an allocation α serving $e \in E_\mathrm{p}$.

$$\gamma^e = (\gamma^e_1, \ldots, \gamma^e_n) \text{ with } \gamma^e_i \in V_\mathrm{a} \tag{2}$$

One resource γ^e_i within such a path is called a *hop*. A *feasible paths* fulfills the following conditions:

$$\gamma^e : \gamma^e_1 = r_s \ \wedge \ \gamma^e_{|\gamma^e|} = r_t \tag{3a}$$
$$\wedge\, \forall 1 \leq i < |\gamma^e| : (\gamma^e_i, \gamma^e_{i+1}) \in E_\mathrm{a} \tag{3b}$$
$$\wedge\, \forall 1 \leq i \leq |\gamma^e| : \gamma^e_i \in \alpha \tag{3c}$$
$$\wedge\, \forall 1 \leq i, j \leq |\gamma^e| \wedge i \neq j : \gamma^e_i \neq \gamma^e_j \tag{3d}$$

Eq. (3a) ensures that the path starts at the resource r_s of the sending process p_s and ends on the resources r_t of the receiving process p_t. Eq. (3b) ensures that there exists an interconnection between all used hops. Eq. (3c) takes care of all used hops being allocated, while Eq. (3d) forbids loops in the path.

To enable the design space exploration process to find all global optima, every path $\gamma^e \in \Gamma^e_{\alpha,\beta}$ has to be evaluable by the exploration. Thus, the used algorithm to perform the routing must ensure that every path has a chance to be chosen:

$$\forall \gamma^e \in \Gamma^e_{\alpha,\beta} : P(\gamma^e) > 0 \tag{4}$$

Note that, e.g., a shortest path algorithm only considers paths with minimal length k, thus, paths $|\gamma^e| > k$ of greater length have a probability $P(\gamma^e) = 0$ and are pruned from being chosen, respectively.

Furthermore, the used algorithm should perform a *fair* routing, i.e., every $\gamma^e \in \Gamma^e$ should have the same probability $P(\gamma^e)$ to be chosen:

$$P(\gamma^e) = \frac{1}{|\Gamma^e|} \tag{5}$$

C. Changing Topologies

The need for a technique that can determine a routing γ while handling different underlying topologies can be explained by an example based on the graph in Fig. 2: Imagine the process p_1 being bound to r_1, p_2 being bound to r_3 and p_3 being bound to r_2, cf. Fig. 2a). The data dependency d_2 from p_1 to p_3 can either be routed directly over *BUS* 1 or over *BUS* 2 and a direct connection between r_3 and r_2. On

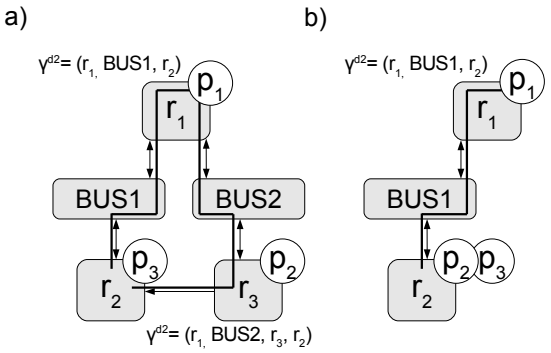

Fig. 2. Implementations of the specification in Fig. 1: Implementation a) offers two paths to establish the data dependency d_2 while implementation b) only offers one possible path, because resource r_3 and $BUS2$ are not allocated.

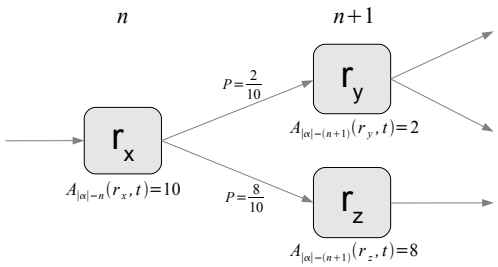

Fig. 3. Sampling for a fair routing in step n. The number of feasible paths A that are still left to reach the target determine the probability of a hop to be taken in $n+1$.

the other hand, process p_2 can also be bound onto r_2 in the first phase of allocation and binding, cf. Fig. 2b). In this case, $BUS2$ as well as r_3 do not have to be allocated anymore and, thus, are not available as hops for a path to pass the data. In current optimization approaches, the hops, i.e., the avail resources are fixed and, thus, used directly to encode the routing in the Evolutionary Algorithm. Hence, a change in the topology leads to a loss of all the genetic routing information that may still be valid and is often already optimized for parts of the topology that are also present in the changed topology.

To overcome this drawback, a genetic encoding independent of the underlying topology as well as a sampling algorithm that enables to calculate feasible paths based on this encoding are introduced in the next section.

V. Path-Sampling in Unknown Topologies

In this section, a novel path sampling algorithm is introduced that is controlled by decision values. These decision values can be genetically encoded and substitute the encoding of the topology of the network embedded system.

A straightforward solution for a routing algorithm fulfilling the fairness property stated in Eq. (5) is randomly choosing a path $\gamma^e \in \Gamma^e_\alpha$. Hence, this algorithm needs to compute the set Γ^e_α explicitly. Unfortunately, the computation complexity of $|\Gamma^e_\alpha|$, which is the number of simple paths of maximum length $l_{\max} = |\alpha| - 1$, is already $\#P$-complete [21]. As a consequence of the calculation of the number of simple paths being that complex, enumerating all simple paths to form the set Γ^e_α is an even harder problem.

A. Path Sampling using Weighted Hops

To avoid the complex computation of Γ^e_α, a constructive heuristic is presented that samples a path from the source to the destination guided by the genetically encoded decision values. At this juncture, the number of feasible paths $A_l(s,t)$ that are still left to reach the target t from the source s in l steps are used to weight each hop. These weights, as well as the genetically encoded decision values, are then used to decide which hop is taken next during the path sampling process.

The principle of the path sampling is illustrated in Fig. 3. Imagine the sampling has come in n steps to the resource r_x. Moreover, it is known that a maximum of $l = |\alpha| - n$ steps are left to reach the target with a loop-free path. The weight of r_x is $A_l(r_x, t) = 10$, i.e., there are 10 feasible paths being left to reach the target. In step $n+1$, there are two resources reachable that can be taken as the next hop: r_y with $A_{l-1}(r_y, t) = 2$ feasible paths and r_z with $A_{l-1}(r_z, t) = 8$ feasible paths left. To ensure a fair routing, r_y, thus, has to be taken with probability $\frac{2}{10} = 0.2$ and r_z with probability $\frac{8}{10} = 0.8$. The decision which hop is taken next is hereby guided by the genetically encoded decision variable h^e_n. The complete sampling approach is outlined in Algo. 2. Based on the weights of each hop and the derived probabilities of each next to be taken, the presented algorithm approach approximates Eq. (5) without the need for the complex calculation and storage of the set Γ^e_α.

B. Genetic Encoding

The used genotype consists of an encoding for the allocation $\alpha \subseteq V_\mathrm{a}$ realized as bit-vectors and of an allocation-independent encoding of the binding $\beta \subseteq E_\mathrm{m}$ by using priority lists [4]. Moreover, the decision values for the path sampling algorithm are encoded using a set $h = \{h^e | e \in E_\mathrm{p}\}$ of vectors $h^e = (h^e_1, \ldots, h^e_{|V_\mathrm{a}|})$. Thus, the sampling of a path for each data dependency $e \in E_\mathrm{p}$ is guided by modifiable a vector of decision values. The initial genotype of each individual is constructed the following:

$$\forall e \in E_\mathrm{p}, 1 \leq i \leq |V_\mathrm{a}| : h^e_i = \mathbb{R}_{[0,1)} \tag{6}$$

The used genetic operators to handle the introduced encoding are the *normally distributed mutation* [17] with a handling for bounded decision values and the so-called *Simulated Binary Crossover* [8].

VI. Weighting Hops using Matrices

In the following, two weighting techniques based on adjacency matrices and matrix multiplication are introduced. These approaches solve the problem of weighting hops according to the number of feasible paths to the target. Furthermore, an enhancement for both algorithms to take the limited capacity of the communication interconnections into account is presented.

Algorithm 2 The sampling algorithm to perform the actual routing from source to destination by using weighted hops. The algorithm serves Eq. (3a) in lines 2 and 3 through determining source and target resource. Until the target is reached, see line 6, every used hop is appended to the path in line 7. Eq. (3b) and Eq. (3c) are ensured via line 10. Line 11 is used to calculate an actual decision value using the weight of the actual hop and the possible next one. The decision which hop to take next is done in line 12 by comparing the actual decision value and the genetic encoded one. If the target is reached, the path is returned.

1: **procedure** SAMPLING(data dependency $e = (p, \widetilde{p}) \in E_\mathrm{p}$, allocation α, binding β)
2: source $s = v \in V_\mathrm{a} \wedge (p, v) \in \beta$
3: destination $t = v \in V_\mathrm{a} \wedge (\widetilde{p}, v) \in \beta$
4: $a = s$
5: $n = 1$
6: **while** $a \neq t$ **do**
7: $\gamma^e_n = a$
8: $\sigma = 0$
9: $l = |\alpha| - n$
10: **for each** $(a, \widetilde{a}) \in E_\mathrm{a}$ with $\widetilde{a} \in \alpha$ **do**
11: σ += $A_{l-1}(\widetilde{a}, t)$ / $A_l(a, t)$
12: **if** $h^e_n \leq \sigma$ **then**
13: $a = \widetilde{a}$
14: break
15: **end if**
16: **end for**
17: $n = n + 1$
18: **end while**
19: $\gamma^e_n = t$
20: **return** routing path γ^e
21: **end procedure**

A. Plain Path Enumeration

The first algorithm called *plain path enumeration* uses an extended adjacency matrix to weight hops. For this purpose, the matrix for the topology α is generated that is available after the allocation has been performed:

$$A = (a_{i,j})_{1 \leq i,j \leq |\alpha|} = \begin{cases} 1, & i,j \in \alpha \wedge ((i,j) \in E_\mathrm{a} \vee i = j) \\ 0, & otherwise \end{cases} \tag{7}$$

Taking A to the l-th power results in a matrix with the number of paths of maximum length l including (self-)loops [1, Sec. 5.9].

$$A_l = A^l \tag{8}$$

The maximum length of a feasible, loop free path calculates as the number of resources minus 1, i.e., $|\alpha| - 1$. Therefore, it is sufficient to calculate and store the matrices $A_1, \ldots, A_{|\alpha|-1}$. Of course, these matrices have to be calculated only once for each α.

B. Self-Loop Free Path Enumeration

Since the plain path enumeration also includes self-loops, short paths are weighted much heavier than long paths. Therefore, an enhancement can be introduced that excludes self-loop. This algorithm is called *self-loop free path enumeration*. For this purpose, the adjacency matrix itself is used:

$$\widetilde{A} = (a_{i,j})_{1 \leq i,j \leq |\alpha|} = \begin{cases} 1, & i,j \in \alpha \wedge (i,j) \in E_\mathrm{a} \\ 0, & otherwise \end{cases} \tag{9}$$

Taking $\widetilde{A}$ to the l-th power results in a matrix $\widetilde{A}^l$ containing the number of feasible paths of exact length l. The matrices used for sampling consisting of feasible paths of maximum length l then calculate as:

$$A_l = \sum_{k=1}^{l} \widetilde{A}^k. \tag{10}$$

Of course, this algorithms still contains loops of length $l > 1$, but eliminating these loops requires, hence, an increasing amount of extra computational effort.

C. Incorporating Capacity Constraints

A drawback of both plain path enumeration and SL-free path enumeration is that routing constraints based on a too low remaining capacity cannot be considered. Thus, topologically feasible paths can still be infeasible due to capacity constraints. The remaining capacity of a communication edge $\widehat{e} \in E_\mathrm{a}$ is defined as follows:

$$c^r_{\widehat{e}} = c_{\widehat{e}} - \sum_{\gamma^e \in \gamma} d_e \cdot x_{\gamma^e, \widehat{e}} \tag{11}$$

With $x_{\gamma^e, \widehat{e}}$ being 1 if the communication edge $\widehat{e} \in E_\mathrm{a}$ is used in a path γ^e and 0 otherwise. Therefore, a feasibility check after the sampling of a path is needed. To overcome this drawback, a modification can be applied to both algorithms. Hereby, an adjacency matrix has to be calculated for every demand d_e. At this juncture, such connections are set to 1 that have enough capacity c^r to serve the demand, while others are set to 0, just like missing connections, respectively:

$$A^* = (a_{i,j})_{1 \leq i,j \leq |\alpha|} = \begin{cases} 1, & i,j \in \alpha \wedge (\widehat{e} = (i,j) \in E_\mathrm{a} \vee \\ & \quad i = j) \wedge d^e \leq c^r_{\widehat{e}} \\ 0, & otherwise \end{cases} \tag{12}$$

$$\widetilde{A}^* = (a_{i,j})_{1 \leq i,j \leq |\alpha|} = \begin{cases} 1, & i,j \in \alpha \wedge (\widehat{e} = (i,j) \in E_\mathrm{a}) \\ & \quad \wedge d^e \leq c^r_{\widehat{e}} \\ 0, & otherwise \end{cases} \tag{13}$$

Using this matrices for the exponentiation, sampling a feasible path also under given capacity constraints, is guaranteed. Hence, the drawback of this modification is a high computational cost that scales in the number of demands.

D. Example

The first sampling step using plain path enumeration for the example in Fig. 1 is shown in Fig. 4. The reachable hops are determined using the matrix A which has a resource order of

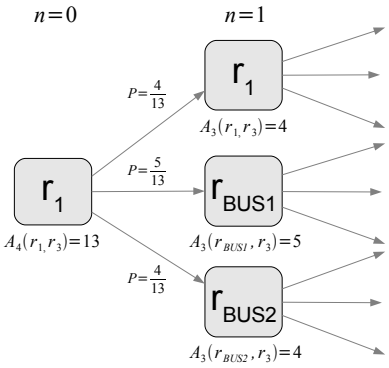

Fig. 4. Example: First sampling step for the routing of demand d_3 from resource r_1 to r_2.

$\{r_1, r_2, r_3, BUS\,1, BUS\,2\}$:

$$A = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{14}$$

The possible next hops are written in bold letters. The weighting of the hops is done by using the matrices A_3 and A_4, with the weights of the hops shown in Fig. 4 written in bold letters:

$$A_3 = \begin{pmatrix} 7 & \mathbf{4} & 3 & 6 & 6 \\ 3 & 4 & 0 & 5 & 1 \\ 4 & 5 & 4 & 4 & 5 \\ 6 & \mathbf{5} & 1 & 7 & 3 \\ 6 & \mathbf{4} & 5 & 4 & 7 \end{pmatrix} \tag{15}$$

$$A_4 = \begin{pmatrix} 19 & \mathbf{13} & 9 & 17 & 16 \\ 9 & 9 & 1 & 12 & 4 \\ 13 & 13 & 9 & 13 & 13 \\ 16 & 13 & 4 & 18 & 10 \\ 17 & 13 & 12 & 14 & 18 \end{pmatrix} \tag{16}$$

The sampling approach can now use the genetic encoded decision values to sample the path. As next hops with a weight of 0 are not considered by the proposed approach, reaching the target resource is guaranteed in a maximum of $|\alpha| - 1$ steps.

VII. Experimental Results

The presentation of the performed experimental results is twofold: The first part deals with a static topology in order to have a better competition of the introduced path sampling algorithms. The second parts describes design space exploration test cases done for the combined topology and routing optimization.

Since the presented weighting algorithms based on matrices do not exclude loops from the paths, an exact algorithm that eliminates all loops from the weighting of each hop has been implemented, too. This so-called *loop free path enumeration* only considers the subset of hops that are reachable for each routing demand. Afterwards, it performs a modified breadth-first search algorithm with exponential time complexity. Hence, this algorithm was only applicable for examples containing a maximum of about 10 to 15 allocated resources.

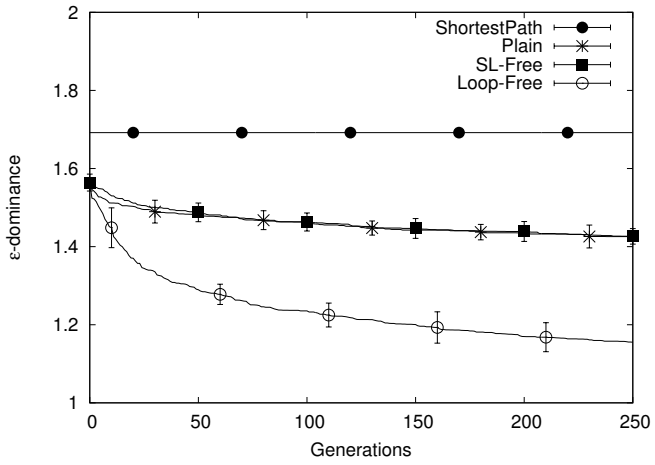

Fig. 5. ϵ-dominance over generations for the test case of the routing only approach in Sec. VII-A.

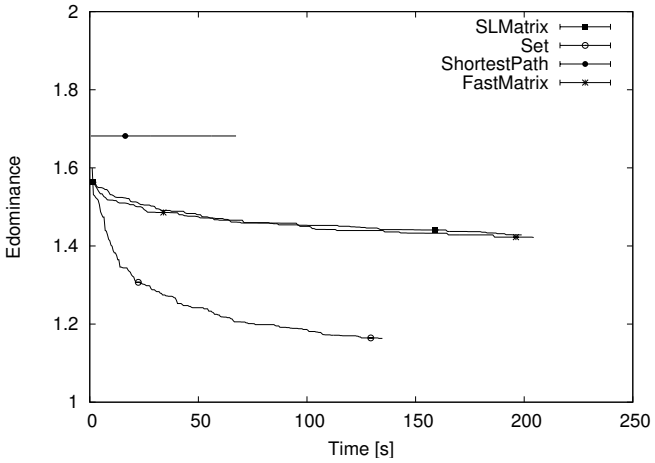

Fig. 6. ϵ-dominance over time for the test case of the routing only approach in Sec. VII-A.

In order to evaluate the quality of the methods, the ϵ-dominance [22] criterion is used. The ϵ-dominance specifies the convergence of a set of solutions A to the set of the Pareto-optimal solutions B, which is approximated as the best solutions of all methods and runs.

$$\mathcal{D}_\epsilon(A, B) = \inf_\epsilon \{b \in B \mid \exists a \in A : a \succeq_\epsilon b\} \tag{17}$$

A smaller ϵ-dominance value corresponds to a better convergence to the Pareto-optimal front.

A. Routing Optimization

To evaluate the influence of the loops that are included in the presented weighting algorithms, several topologies with 5 up to 40 network resources and a fixed binding were generated. The probability of two resources being linked was specified as a ratio. The ratio for the generated test-cases was varied

	Fig. 5 time[s]	Fig. 7 time[s]
ShortestPath	68	132
Plain	204	147
Plain*	–	294
SL-Free	198	177
Loop-Free	134	144

TABLE I

AVERAGE TIME CONSUMPTION OF THE EXPLORATIONS USING THE DIFFERENT PATH ENUMERATION STRATEGIES.

between 0.2 and 1. For each test-case class, 10 instances were created with each being explored 10 times with a population size of 100 individuals. Furthermore, two times the number of resources routing demands were generated for each instance. For the evaluation of a found solution, three objectives with uniformly distributed values were used.

As an example, the results for 10 resources and a ratio for the degree of resources of 0.5 which is representative for all test cases and enables a comparison to the exact loop free algorithm is presented. In these test cases, routing constraints were neglected, such that the capacity constraints were not incorporated into the plain and SL-free path enumeration strategies. Fig. 5 shows that all presented algorithms clearly outperform the shortest path approach. Noticeable is that eliminating the self-loops with the SL-free path enumeration does not improve the results. Regarding only these test cases, the exact loop free path enumeration is by far superior to the matrix algorithms. Even when considering the time consumption, shown in Tab. I, the loop free path enumeration performs better than the matrix routers. This behavior was expected, since the loop free path enumeration eliminates all loops and therefore allows long, but in some objectives optimal paths to be taken with the same probability.

B. Unified Topology and Routing Optimization

The design space exploration that combines topology and routing optimization was also carried out on the same topologies as in the previous Sec. VII-A. Hence, for each example, two times the number of resources processes were generated in the problem graph, each having a chance of 0.25 of being bindable onto a resource. The connectivity of the problem graph leads to a number of routing demands of about 1.5 times the task count. Again, 10 examples of each class were generated and explored 10 times with a population size of 100 individuals. The considered objectives were area cost, power consumption, and reliability [18]. Representative for all experiments, the results for 20 resources, 40 tasks, and about 60 routing demands are shown in Fig. 7. A little surprising, the loop free path enumeration had the worst results of all methods used. This may be due to the fact that good values for the objectives area cost, power consumption, and reliability are often correlated with a low number of hops taken. Hence, the loop free path enumeration tends to find many relatively long paths. The assumption of a correlation of the objectives

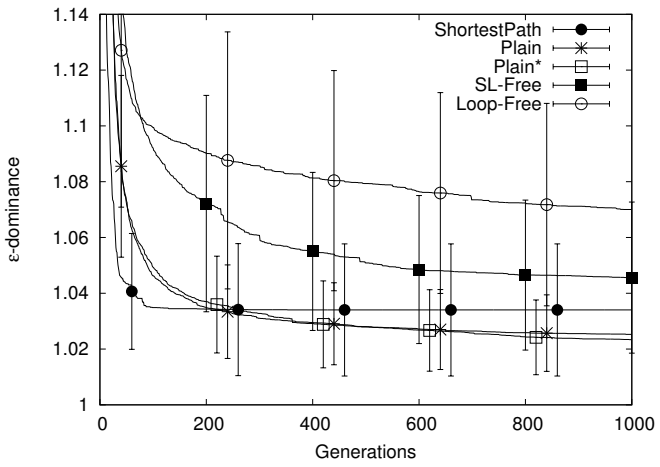

Fig. 7. ϵ-dominance over generations for the test case of the unified optimizing approach in Sec. VII-B.

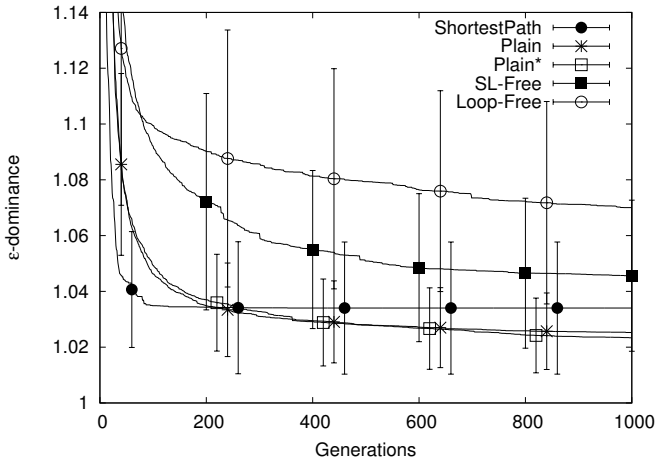

Fig. 8. ϵ-dominance over time for the test case of the unified optimizing approach in Sec. VII-B.

with short paths seems reasonable as also the SL-free path enumeration performs badly in the performed test-cases. The deviation shows that the shortest path approach may perform good or bad, depending on the explored example. Hence, its results are less reliable than the ones achieved by the introduced path sampling approaches. The best performance with a relatively low standard deviation had the plain path enumeration as well as the plain* path enumeration. Tab. I shows that the plain* path enumeration needs a huge amount of extra time that the plain path enumeration can invest on punishing infeasible paths instead of avoiding them, thus achieving results of the same quality.

VIII. CONCLUSION AND FUTURE WORK

In this paper, an approach for a unified multi-objective topology and routing optimization using Evolutionary Algorithms was presented. Therefore, a new algorithm for multi-objective routing with a genetic encoding independent of the underlying topology has been introduced. The performed ex-

perimental evaluation shows the effectiveness of the proposed approach as well as more certain results compared to a common heuristic of using shortest path routing. In future work, a further development of the presented algorithms to work faster and more appropriate for the combined optimization approach is aspired. One attempt could be bounding the maximum length $l_{\max}$ of a path to l_{b}, with $0 \leq l_{\mathrm{b}} \leq l_{\max}$, which seems reasonable when dealing with latency constraints or communication over buses only, and include this parameter in the evolutionary optimization. This could highly improve the performance of the presented algorithms by pruning the search space.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] A. F. R. Araujo, C. Garrozi, A. R. G. A. Leitao, and M. M. J. Gouvea, "Multicast routing using genetic algorithm seen as a permutation problem," in *Proceedings of the 20th AINA*, Washington, DC, USA, 2006, pp. 477–484.

[3] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA - A Platform and Programming Language Independent Interface for Search Algorithms," in *EMO*, vol. 2632, Faro, Portugal, Apr. 2003, pp. 494–508.

[4] T. Blickle, J. Teich, and L. Thiele, "System-level synthesis using Evolutionary Algorithms," *J. Design Automation for Embedded Systems*, vol. 3, no. 1, pp. 23–58, January 1998.

[5] B. Boffey, "Multiobjective routing problems," vol. Top3, no. 2, pp. 167–220, 1995.

[6] CAN, "Controller Area Network," http://www.can.bosch.com/.

[7] J. Crichigno and B. Barn, "A multicast routing algorithm using multiobjective optimization." in *Telecommunications and Networking - ICT 2004*, 2004, pp. 1107–1113.

[8] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[9] B. K. Dwivedi, A. Kumar, and M. Balakrishnan, "Automatic synthesis of system on chip multiprocessor architectures for process networks," in *Proceedings of the 2nd conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, 2004, pp. 60–65.

[10] C. Erbas, S. C. Erbas, and A. D. Pimentel, "A multiobjective optimization model for exploring multiprocessor mappings of process networks," in *Proceedings of the 1st conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, 2003, pp. 182–187.

[11] FlexRay, http://www.flexray.com/.

[12] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of the 3rd conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, 2005, pp. 75–80.

[13] C. Haubelt, "Automatic Model-Based Design Space Exploration for Embedded Systems – A System Level Approach," Ph.D. dissertation, University of Erlangen-Nuremberg, Germany, Berlin, July 2005.

[14] R. Kumar and N. Banerjee, "Multicriteria network design using evolutionary algorithm." in *GECCO*, 2003, pp. 2179–2190.

[15] LIN-Subbus, "LocalInterconnect Network," http://www.lin-subbus.org/.

[16] M. Palesi and T. Givargis, "Multi-objective design space exploration using genetic algorithms," in *Proceedings of the tenth international symposium on Hardware/software codesign (CODES)*, 2002, pp. 67–72.

[17] I. Rechenberg, *Evolution Strategy: Optimization of Technical Systems by Principles of Biological Evolution.* (In German) Stuttgart: Frommann-Holzboog, 1973.

[18] T. Streichert, M. Glaß, C. Haubelt, and J. Teich, "Design space exploration of reliable networked embedded systems," *Journal on Systems Architecture*, vol. 53, no. 10, pp. 751–763, 2007.

[19] T. Streichert, C. Haubelt, and J. Teich, "Multi-Objective Topology Optimization for Networked Embedded Systems," in *Proceedings Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS 2006)*, Samos, Greece, July 2006, pp. 93–98.

[20] TTP, "Time Triggered Protocol," http://www.TTTech.com/.

[21] L. G. Valiant, "The complexity of enumeration and reliability problems." *SIAM Journal on Computing*, vol. 8, pp. 410–421, 1979.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review." *IEEE Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.