Evolutionary Routing Strategies for Automotive Networks

Martin Dohr, Bernd Eichberger
Institute of Electronics, Graz University of Technology, Austria

Abstract— *The design of efficient communication networks is a challenging task for modern vehicle development. Due to novel technologies and new degrees of freedom in network design, the decision on a bus topology has a severe impact on the overall system cost and performance. In this work, we propose a topology and routing optimization using Evolutionary Algorithms and problem-specific encoding. Our contribution includes a guided topology mutation operator which outperforms standard random mutation. Further, we propose two routing operators for usage during the optimization process and compare their effectiveness on a network application taken from a series vehicle.*

Keywords: Evolutionary computation; communication networks; design optimization; routing;

1. Introduction

Upcoming developments in the automotive industry will have a severe impact on in-vehicle communications. Such developments include advanced driver assistance systems like lane departure warning, night vision or traffic sign recognition. Those systems have in common, that complex sensors such as cameras, RADAR or LIDAR have to be deployed in remote locations within car. In order to convert the raw sensor data into usable information for the driver, sophisticated evaluation algorithms have to be applied. System designers have always been mapping such computational intensive tasks onto central Electronic Control Units (ECUs) for cost reasons. Another argument for centralized processing is, that upcoming functionalities will have a interconnected architecture and rely on multiple distributed sensors as input data sources. However, in contrast to present features, those new developments and sensors will also have much higher communication requirements which cannot be satisfied by well established automotive bus systems like CAN [1]. For example, the raw data transfer of RADAR and LIDAR sensors are comparable to monochrome or colored video stream. In comparison to that, current automotive features such as cruise control or automated headlamp leveling only require a fractional amount of that bandwidth.

There are several solutions to meet the new communication demands in the car. First implementations of rear view cameras were connected by a separate shielded twisted pair or coaxial wire using analog data transmission. While this is cost efficient for a single data source, the cabling effort for several distributed sensors makes this solution impractical for modern automotive architectures. Improved bus systems like MOST [2], FlexRay [3] or Ethernet [4] are better suited for such high bandwidth demands and also satisfy the stringent reliability and real time requirements of automotive applications.

Another aspect are new architecture paradigms to manage the increased complexity of interconnected features. Architectures like AUTOSAR [5] define an interface to separate functional software from the underlying operating system and hardware, thus allowing modular design of software components (SWCs). The advantages of this separation are that software components are now more independent from their location of execution and can be mapped onto different ECUs within the vehicle. At the same time, testability and re-useability of components increases because of the standardized interface and even online relocation of software execution within the car is possible.

However, the architecture and design process of a network is becoming more complex because of such recently introduced degrees of freedom in software mapping. In [6], the authors propose a holistic network architecture process using Evolutionary Algorithms (EAs) and problem-specific encoding. In this paper we will focus on methods to improve the topology and routing optimization of the mentioned approach.

The rest of this paper is structured as follows: Section 2 presents related publications concerning topology optimization using evolutionary methods with a further focus on automotive applications. Our system model and the phases of optimization are explained in Section 3. A more detailed description of our topology mutation and routing operators is then given in Section 4 before presenting our test scenario and results in Section 5. Finally, we give a conclusion of the current work and next plans in Section 6.

2. Related Work

Network topology optimization has been an ongoing research topic for decades as summarized in [7]. Most works focus on multiobjective heuristics to solve such problems [8] with objectives like network cost or message delay. The performance is then verified on realistic traffic models as in [9]. Recent works [10] utilize objective-guided genetic operators in combination with MOEA/D to incorporate problem-specific knowledge into the optimization process.

Several authors have also studied special attributes of automotive networks and how to optimize this specific application. In [11], a repeated-matching method combined with

simulated annealing is proposed to optimize task allocation and network assignment. A similar problem was solved in [12] using a sophisticated system model and multiobjective EAs. A hierarchical partitioning is used in [13] in combination with local heuristics for the gateway placement task, a sub-problem which is not addressed in most publications. The authors of [14] propose a combination of Integer Linear Program (ILP) and EAs for topology and routing optimization but also require a given architecture in form of nodes, buses and gateways.

3. Optimization Work Flow and System Model

Our optimization framework distinguishes between two phases, application mapping and signal routing. Both phases run adapted implementations of the SPEA2 algorithm [15] which are extended by application specific network encoding and custom operators. Each phase optimizes the network with respect to different goals by utilizing specific fitness functions and operators suited for the task at hand. The overall work flow is depicted in Figure 1. Our work builds on the 'Metaheuristic Algorithms in Java' framework presented in [16], which was chosen due to the very modular and extendable design.

The aim of the first phase is to map all software components onto ECUs while keeping communication between the nodes at a minimum. On the other hand, the deployment has to respect the processing capabilities of each ECU and safe costs by optimal utilization for each node. Since those goals are contradicting, the result of this phase is a Pareto set of possible software mapping solutions. In our encoding, we call such a mapping candidate an ApplicationMatrix. Therefore, each solution in phase one consists of a common functional description and set of nodes, combined with a unique ApplicationMatrix, which defines the mapping of each SWC onto a node. Further details on the used operators will be published in [17] and are not the focus of this paper. The rest of this section is mainly concerned with input objects and variable encoding for the second optimization phase.

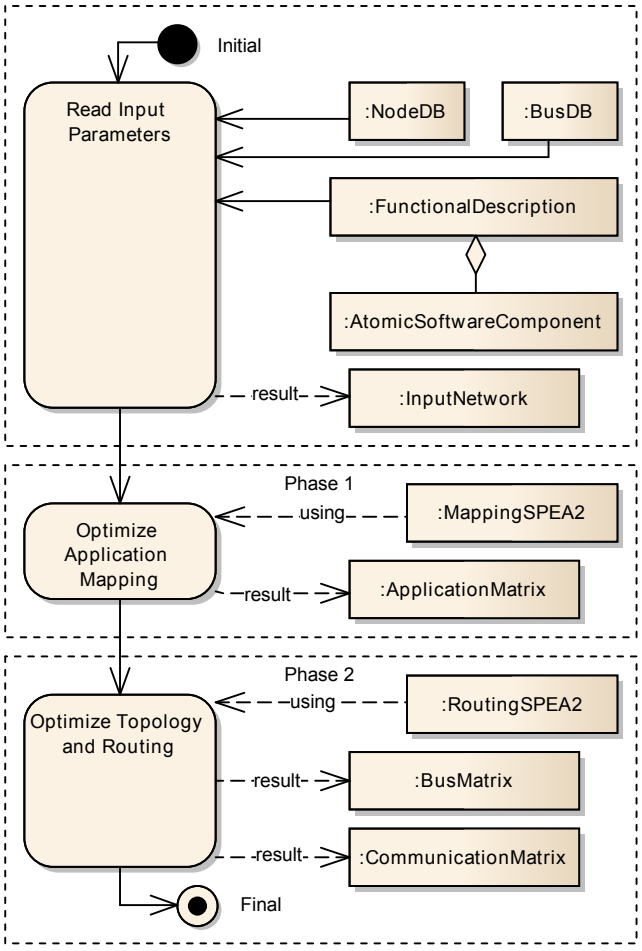

Fig. 1: Workflow of the proposed framework showing input objects and the two phases of optimization

3.1 Input Objects

In order to explain the goals of phase two it is sufficient to know, that the input from phase one is a set of one or many Pareto optimal mapping solutions for a given set of nodes and functional description. The goal of this phase is to create a feasible network topology where all nodes are connected via bus systems or gateways and each communication signal can be routed over this topology. The input parameters shall be defined as follows:

- *NodeDB:* A list of active nodes N where each node $n \in N$ corresponds to an ECU with given coordinates within the vehicle. Note that a model of monetary costs for each node is only required in the first phase of optimization. In phase two, it is sufficient to estimate the additional costs of connecting a node to a bus system which is modeled as bus coupler costs.
- The *FunctionalDescription* is a logical network of all software components C and their corresponding communication signals S. Each signal $s_i \in S$ consists of a set of one or more receiving SWCs C_{s_i} and the corresponding bandwidth requirement in kbit/sec. Since our framework is designed to be used in a very early stage of product development, we provide a rough bandwidth estimation instead of detailed transmission deadline constraints which are generally not known at this time.
- *BusDB:* A database of all available bus systems B with estimated cost factors for bus couplers and wiring effort. Furthermore, each bus system $b \in B$ is defined by a maximum transfer capacity in kbit/sec. This capacity is usually set between 30% and 70% of the theoretical maximum at this stage of development to be prepared

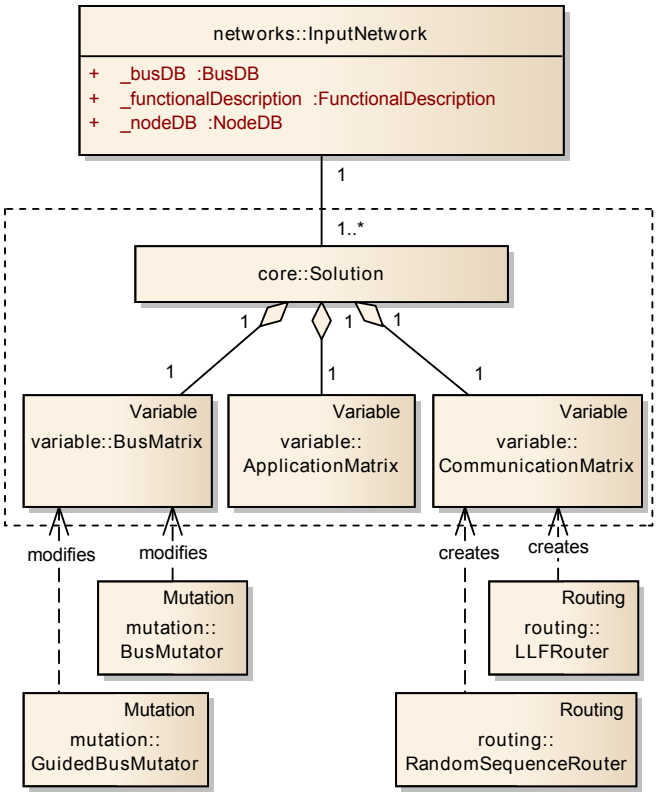

Fig. 2: Diagram of all related input objects, variables and operators

for changes in communication requirements. A detailed schedulability analysis, as presented in [18], is not feasible at this stage of development.

- One or several *ApplicationMatrix* representations of possible solutions from phase one.

3.2 Variable Encoding

A complete solution candidate in phase two consists of common input parameters, a mapping solution from the previous phase and two new variables representing topology and signal routing. First, we define a variable called BusMatrix to model the connections between nodes and buses. The dimension is $|B| \times |N|$ with each entry stating whether the node is connected to this bus system or not. The network topology is therefore defined by this variable with the exception of gateway functionalities. The representation can be altered by mutation operators which is the usual practice in evolutionary algorithms. Since the algorithm can work with several distinct mapping solutions in parallel, a crossover operation would not be suitable in this phase.

It should be noted, that a BusMatrix does not provide a feasible network until a signal path has been found for each communication signal. This is done by a routing operator and results in a CommunicationMatrix, which represents all signal paths and gateways within the network. The routing operator uses a BusMatrix as input but can also add connections to repair infeasible communication paths. Further, the routing procedure can add gateway functionalities onto nodes if needed. We consider gateways as objects derived from software components and only allow a mapping on certain nodes in the system. Those mappings can be created by the router or by a distinct gateway mutation operator. During the routing process, the CommunicationMatrix is filled with signal paths and bandwidth information for all used bus systems accompanied by a set of mapped gateways G and their respective routing tables. In order to preserve feasibility during this process, we define two constraints for the routing algorithm.

1) Signals must not be routed over a bus system if the resulting bandwidth would exceed the maximum bus capacity.
2) To limit complexity and transmission time, a signal route over two or more gateways is not allowed. This is also standard practice for current vehicle networks which utilize one central gateway for all bus systems.

A overview of all mentioned objects is depicted in Figure 2.

3.3 Fitness Functions

The goal of the routing phase is to find feasible networks utilizing availiable bus systems in a cost efficient way. This implies the following statements.

1) Each bus system shall have optimal bandwidth utilization.
2) A balance between faster (more expensive) and cheaper bus technologies has to be found.
3) Wire lengths have to be minimized as the cabling harness is a significant cost factor in modern vehicles.
4) The usage of gateway components doubles the bandwidth requirement for a signal since it has to be transmitted over two bus systems to reach all receiving software components.

Based on those statements we define three objectives to measure the fitness of a solution.

System cost defines the hardware costs for a given topology. The bus coupler cost $BCC(b)$ models the expense of adding a node n to a bus system b. The set $N(b)$ is the set of all nodes that are connected to the bus system b.

$$SystemCost = \sum_{b \in B} BCC(b) \times |N(b)|$$

Wiring cost represents the length of the cable harness. We approximate this length for every bus system as the sum of Manhattan distances between all nodes connected the bus. The Manhattan distance is defined as the path between two nodes, when only moves in direction of the Cartesian axes are allowed.

$$dist(n_1, n_2) = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

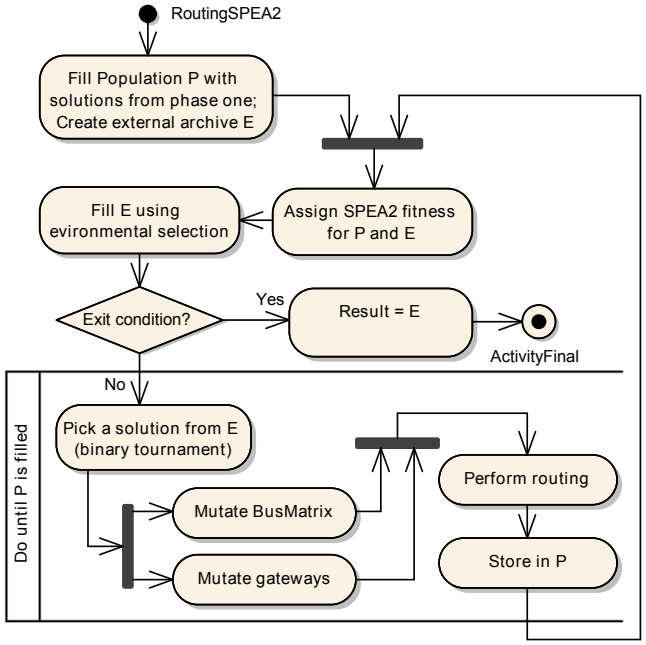

Fig. 3: The routing algorithm with SPEA2 selection methods and custom mutation and routing operators

For certain bus systems like MOST, the wire length can by multiplied by an additional cost factor to reflect the more expensive installation effort for optical cables.

Bus utilization is modeled by a fuzzy set where a consumed bandwidth between 30% and 70% of the maximum speed is considered as optimal load. Those values may vary depending on the bus system in use and can be adapted from 'best practices' in automotive engineering. If a bus utilization is out of this range, we model a penalty value based on the bandwidth mismatch and priority of the bus system. Due to the priority, we can force the optimization process to utilize more expensive bus systems better than cheap ones. The overall fitness function is then the sum over the resulting penalty values for all used bus systems.

4. Mutation and Routing Strategies

As stated before, the algorithm uses mutation operators on a BusMatrix and then creates a CommunicationMatrix by applying a routing operator. The procedure is based on the SPEA2 [15] and employs unaltered fitness assignment and selection implementations from [16]. The principal work flow is depicted in Figure 3.

4.1 BusMatrix and Gateway Mutation

We have developed two mutation operators for the BusMatrix, a random mutation and an advanced guided operator. The random implementation can be compared to a simple bit flip mutation, connecting or disconnecting a node from a bus with a certain probability. In contrast to that, the guided mutation prefers to connect nodes to bus systems, whose bandwidth utilization is below optimal values. Consequently, nodes have a higher probability of getting disconnected from bus systems with higher communication load. Algorithm 1 states the procedure for guided bus mutation. It is important to limit the guiding factor to very small values in order to avoid overshooting.

Algorithm 1 Guided BusMatrix Mutation

```
1:  p ← MutationProbability
2:  for all Bus Systems b in B do
3:    u ← BusUtilization(b)
4:    for all Nodes n in N do
5:      if (n connected to b and u == HIGH) or
        (n not connected to b and u == LOW) then
6:        p_n ← p × 1.05
7:      else
8:        p_n ← p
9:      end if
10:     mutate connection with probability p_n
11:   end for
12: end for
```

The mapping of gateway SWCs is also altered by an distinct mutation. It is again a random bit flip mutator method but with much lower probability of execution. Like all other genetic operators, the presented mutation functions share the same interface and could be interchanged during runtime of the optimization.

4.2 Routing Strategies

We compared 2 routing operators with different characteristics. The 'Largest Load First Router' (LLFRouter) is specified by a strict deterministic behavior. This means, given the same BusMatrix and ApplicationMatrix as input, the router will always create the same CommunicationMatrix as output. The routing always starts with the signal representing the highest bandwidth requirement and ends with the signal having the lowest. As a consequence, all evolutionary logic is done during the mutation of the BusMatrix. In contrast to that, we also propose a 'Random Sequence Router' (RSRouter), where the order of routed signals does not follow a certain rule. The results are influenced by the given bus topology but can vary depending on the router logic.

Independent of the chosen operator, the router follows 4 steps to find a feasible path on the current topology for each signal:

1) Find a direct connection between source and destination.
2) Find an available gateway connection.
3) Create a gateway connection without adding new bus couplings.
4) Create a new direct connection.

Depending on the current BusMatrix, a step could find more than one feasible path to route a signal. For this case, we define selection schemes which are set globally for the optimization process:

- *BEST_FIT:* The router chooses the bus system in such a way, that the added signal improves the overall bus utilization. A simple local search heuristic was designed to find this system in a time-efficient way.
- *CHEAPEST:* The cheapest available bus system is always chosen.
- *RANDOM:* The router randomly chooses a path. Note that this will force the router into non deterministic behavior.

5. Experimental Results

We evaluated both routing operators using each mutation strategy and path selection schemes defined above. For each operator setup, we calculated the hypervolume indicator or S-metric [19] and generational distance [20] to compare the resulting Pareto fronts after 20.000 evaluations. Since the true Pareto front for the test problem at hand is not known, we estimated it as the global Pareto front of all evaluation runs.

5.1 Input Network

Our test network consists of 15 ECUs executing 284 atomic software components. The overall communication load created by 390 signals equals 281 kbit/sec. Roughly 30% of those signals have more than one receiving software component, for example network management and diagnostic functions. The structure of this network was generated from a middle class series vehicle but bandwidth requirements were greatly increased as to utilize more bus systems and therefore enable a broader spectrum of possible solutions.

5.2 Results

The resulting performance indicators are listed in Table 1 with the best 3 values for each indicator highlighted in gray. The random sequence router runs slightly faster because the bandwidth sorting algorithm in LLF has to be performed for each routing execution. The comparison shows that guided BusMatrix mutation clearly outperforms its random counterpart without a significant impact on execution time. In fact, the fastest runs were also achieved using guided mutation. An exemplary Pareto plot of 3 different operator setups and the global Pareto front is depicted in Figure 4. It shows the trade off between low system costs versus cheap wiring effort and well utilization of bus systems. The LLF router with guided mutation and random path selection has the best overall performance as it covers both parts of the front equally. Only the combination 'RSR / guided mutation / best fit path selection' is closer to the solutions with low system costs but neglects a lot of solutions in the Pareto front.

Table 1: Results for different operator setups after 20.000 evaluations

Router	Mutation	Path	HV	GD	Exec. time [ms]
RSR	Random	Random	0.09356	0.452851	25336
RSR	Random	Cheapest	0.12951	0.095245	26755
RSR	Random	Best Fit	0.35134	0.062998	25414
RSR	Guided	Random	0.46237	0.021208	24048
RSR	Guided	Cheapest	0.19250	0.016120	24946
RSR	Guided	Best Fit	0.19881	0.005960	25304
LLF	Random	Random	0.01778	0.323714	27539
LLF	Random	Cheapest	0.12510	0.050040	26927
LLF	Random	Best Fit	0.37797	0.035734	29095
LLF	Guided	Random	0.51932	0.008477	27354
LLF	Guided	Cheapest	0.18687	0.010839	29407
LLF	Guided	Best Fit	0.19769	0.006498	30078

6. Conclusion

This paper proposes advanced routing methods for evolutionary network optimization. Our work is based on the framework presented in [6] with focus on the application-specific encoding. The framework is designed to optimize the application mapping and topology in the context of in-vehicle communication. Since this is done in a 2-phase approach, our proposals can focus on the topology and routing with a given set of solutions from the mapping process. First, we developed a guided topology mutation operator to even out bandwidth utilization between different bus systems. Secondly, we propose two routing operators where one shows deterministic behavior while the other is non-deterministic by design. As last input, we chose between three different strategies to find feasible communication paths within the network. After evaluating all operator setups, we find that deterministic routing performs better in most cases when combined with guided topology mutation. Further, this guidance does not significantly prolong the execution time of the overall optimization. However, due to the test scenario taken from a series vehicle, it is not clear if the results can be extended to networks of different sizes. In order to ensure stable behavior for other applications, our next goal is to develop generic test cases for a variety of target applications.

References

[1] Robert Bosch GmbH. (2012) CAN - Controller Area Network. [Online]. Available: http://www.semiconductors.bosch.de/en/ipmodules/can/can.asp

[2] MOST Cooperation. (2012) MOST - Media Oriented Systems Transport. [Online]. Available: http://www.mostcooperation.com

[3] FlexRay Consortium. (2012) FlexRay Communications System. [Online]. Available: http://www.flexray.com

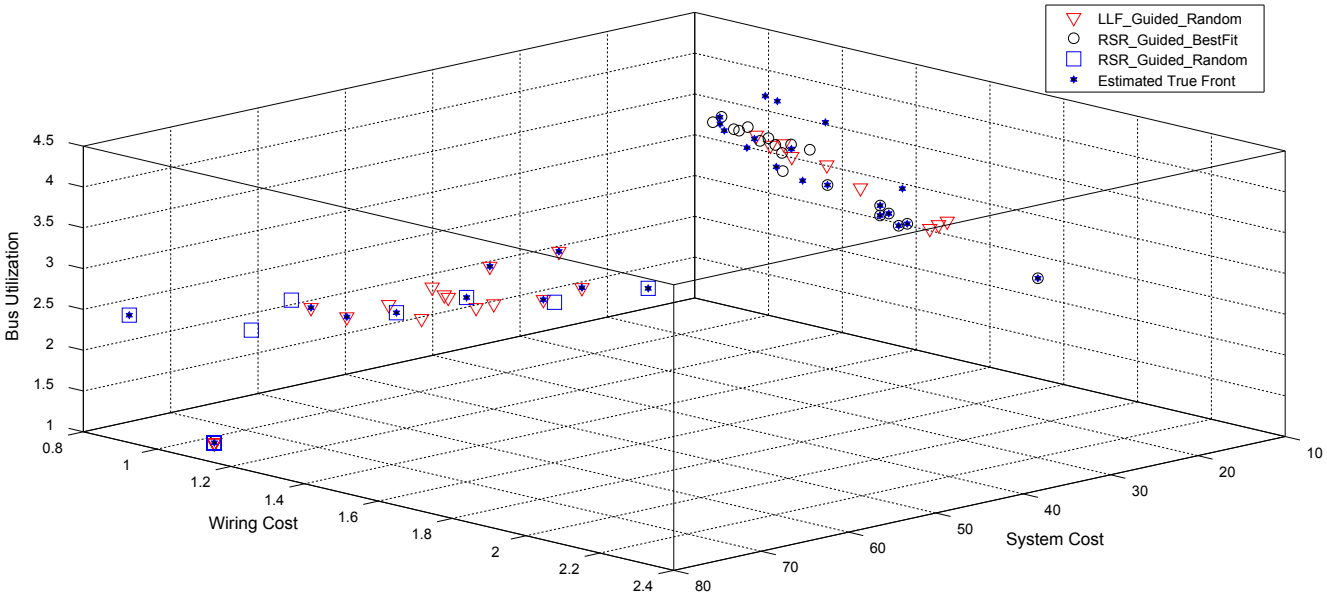

Fig. 4: Pareto plot of 3 operator setups and the estimated global front

[4] H.-T. Lim, L. Volker, and D. Herrscher, "Challenges in a future ip/ethernet-based in-car network for real-time applications," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, june 2011, pp. 7 –12.

[5] AUTOSAR Consortium. (2012) Automotive Open System Architecture. [Online]. Available: http://www.autosar.org

[6] M. Dohr and B. Eichberger, "An application-specific approach in automotive network optimization," in *Proceedings of the 2012 international conference on genetic and evolutionary methods*, jul 2012, pp. 62–67.

[7] M. Abd-El-Barr, "Topological network design: A survey," *Journal of Network and Computer Applications*, vol. 32, no. 3, pp. 501 – 509, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480450800101X

[8] R. Kumar and N. Banerjee, "Multiobjective network topology design," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5120–5128, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2011.05.047

[9] N. Banerjee and R. Kumar, "Multiobjective network design for realistic traffic models," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 1904–1911.

[10] W. Peng and Q. Zhang, "Network topology planning using moea/d with objective-guided operators," in *Parallel Problem Solving from Nature - PPSN XII*, ser. Lecture Notes in Computer Science, C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds. Springer Berlin Heidelberg, 2012, vol. 7492, pp. 62–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32964-7_7

[11] S. Kim, E. Lee, M. Choi, H. Jeong, and S. Seo, "Design optimization of vehicle control networks," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 3002 –3016, sept. 2011.

[12] T. Streichert, C. Haubelt, and J. Teich, "Multi-objective topology optimization for networked embedded systems," in *Embedded Computer Systems: Architectures, Modeling and Simulation, 2006. IC-SAMOS 2006. International Conference on*, 2006, pp. 93–98.

[13] R. Moritz, T. Ulrich, and L. Thiele, "Evolutionary exploration of e/e-architectures in automotive design," in *Proceedings of the International Conference on Operations Research*, Zurich, Switzerland, 2011, pp. 361–366.

[14] M. Lukasiewycz, M. Glass, C. Haubelt, J. Teich, R. Regler, and B. Lang, "Concurrent topology and routing optimization in automotive network integration," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, june 2008, pp. 626 –629.

[15] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*. CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[16] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[17] M. Dohr and B. Eichberger, "Guided mutation strategies for multiobjective automotive network architecture," March 2013, currently under review.

[18] R. I. Davis and A. Burns, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Refuted, Revisited and RevisedŤ. Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[19] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study." Springer, 1998, pp. 292–301.

[20] D. Van Veldhuizen and G. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, 2000, pp. 204–211 vol.1.