Search Space Reduction for E/E-Architecture Partitioning

Andreas Ettner

Robert Bosch GmbH, Corporate Reasearch,
Robert-Bosch-Campus 1, 71272 Renningen, Germany
andreas.ettner@de.bosch.com

Abstract. As the design of electrical/electronic (E/E)-architectures is becoming more complex, multi-objective optimization algorithms such as evolutionary algorithms (EAs) have been proposed for generating resource optimized architectures. In this paper we extend existing approaches by excluding infeasible solutions from the search space and thereby enhance the quality and runtime behavior of the optimization.

Keywords: E/E-Architecture Partitioning, Design Space Reduction, Evolutionary Algorithm

1 Introduction

Due to the introduction of new safety and comfort functions related to advanced driver assistance, highly automated driving, and car-to-x connectivity, the number and interconnections of functions in vehicles has been growing over the last decades and is going to grow further over the next years. As a result, distributed vehicle system architectures consisting of heterogeneous computing resources become more complex and power consuming. Thereby, also the complexity of the allocation task problem is growing, which is defined as assigning functions to Electronic Control Units (ECUs) while fulfilling various design constraints, such as safety requirements and resource restrictions (see Figure 1).

In recent years, several optimization methods with the aim of supporting engineers in power and resource aware design of E/E-architectures have been proposed. While Walla [6] presented a mixed-integer linear programming (MILP) approach to optimize function partitioning with respect to energy efficiency, EAs have proven useful for concurrent optimization of multiple objectives, such as performance, cost, and reliability [1], [3], [4]. However, when increasing the number of design constraints, EAs might fail in producing feasible solutions and in converging toward a global maximum. Common approaches to overcome this problem have been compared by Moser [5] and the results showed that repairing infeasible solutions is superior to penalizing and constrain-dominance methods. Yet, the search space still contains all infeasible solutions for each of which the repair function would be invoked. In order to overcome this drawback and to decrease the amount of infeasible solutions, we present an approach to reduce the search space in advance of the optimization run. Thereby, we enhance the quality and runtime behavior of the optimization.

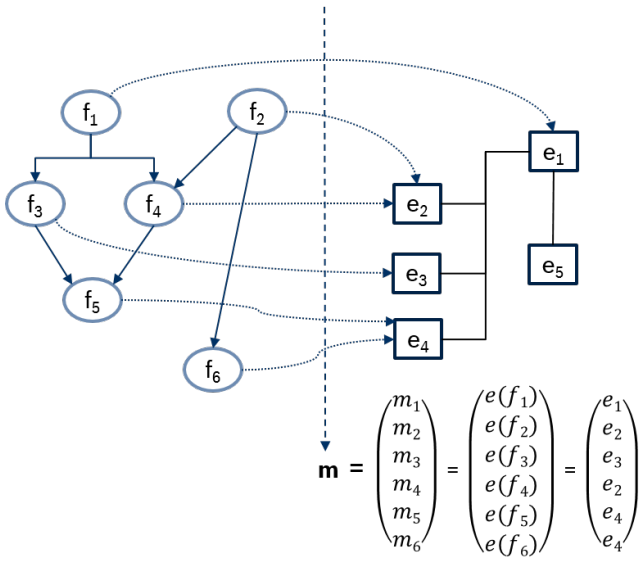

Fig. 1. Function Allocation

2 Concept

2.1 Models and Representation

Figure 1 exemplary shows the two input models to the optimization. The function model consists of a set F of functions with inter-dependencies being represented by directed edges. The architecture is modelled by a set E of ECUs connected by bus structures. Properties describe the available resources on the ECUs and the resource demands as well as functional requirements of the functions. For example, the computing requirements for each function are represented by r_f and the available computing resources for each ECU by r_e, respectively.

The allocation of vehicle functions to the ECUs is represented by a mapping vector m indicating for each function f_i the corresponding ECU identifier e_j. By default, each function can be allocated to each ECU, such that the set of all possible ECUs for f_i is $M = \{e_1, ..., e_J\}$ and the number of possible solutions is $S = |E|^{|F|}$.

2.2 Objective Functions and Constraints

In our approach, we optimize three objective functions: network communication, safety, and the collection of functions with certain properties on a minimum number of ECUs. As the search space reduction is independent of the objective functions, we will concentrate on the constraints in the following:

- Location constraints either define a set of ECUs $C_{I,i}$ - where function f_i will be mapped to one element of this set - or exclude a set of ECUs $C_{E,i}$.

- Colocation constraints either forbid two functions to be allocated to the same ECU ($C_{ban} : e(f_i) \neq e(f_j)$) or force two functions to reside on the same ECU ($C_{force} : e(f_i) = e(f_j)$).
- Some functions demand certain components or functional requirements reqf, which have to be provided by the corresponding ECU (req_e). Therefore, function f_i can only be allocated to one ECU of the set $C_{req,i} = \{e | req_e \geq req_f\}$.
- Some resources, such as the memory and computing units, are shared among all functions allocated to the ECU. Therefore, $r_{e,j} \geq \sum r_{f,i}$ with $(e(f_i) = e_j)$ has to hold for each ECU.

2.3 Genetic Algorithm

For the optimization, we use the NSGA-II algorithm presented by Deb [2]. Its structure is shown in Figure 2. During initialization, a population of solutions is generated by assigning one element of set M to each of the functions resulting in one mapping vector for each solution. Afterwards, the population iteratively improves by creating new solutions, evaluating these solutions with regard to the constraints and the objective functions (see Fig. 3), and finally selecting the best solutions for the next generations based on Pareto-optimality.

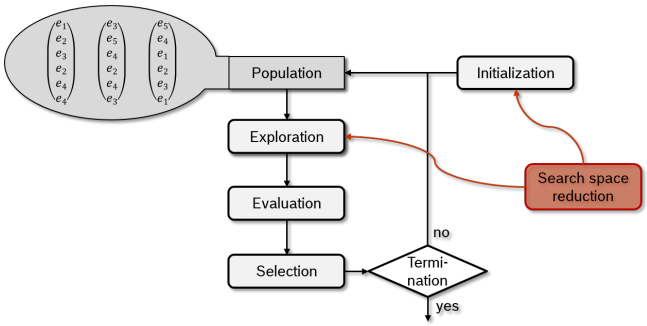

Fig. 2. Structure of Evolutionary Algorithm

New solutions are generated by variation operators such as mutation, which assigns an element of set M to a random number of functions. Thereby, a huge amount of solutions violating the constraints defined in chapter 2.2 might be generated that would have to be either repaired or discarded. As an approach to reduce this number, we reduce the sets M_i for each function f_i in advance of the optimization by analyzing the design constraints and use those sets during initialization and mutation to generate new individuals.

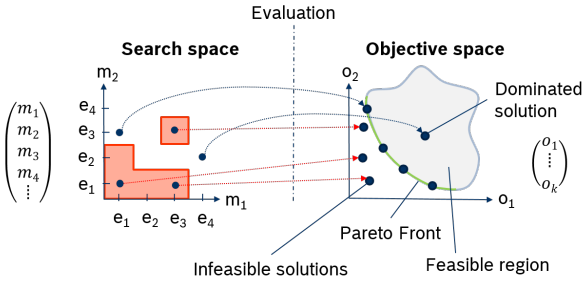

Fig. 3. Evaluation of individuals

2.4 Search Space Reduction (SSR)

Figure 4 presents the SSR for each function. In a first step, the sets Mi are initialized with either the ECUs defined in $C_{I,i}$ (1a) or, in case $C_{I,i}$ is not defined, with all ECUs from M (1b). Afterwards, M_i is reduced by $C_{E,i}$ (2), by the ECUs that do not fulfil the constraints on functional requirements (3), and by the ECUs that do not provide sufficient resources for function f_i (4). Step (5) ensures that the sets M_i and M_j of two functions to be forced together contain the same ECU elements. Finally, if functions to be banned from f_i are already allocated to a certain ECU, this ECU is removed from M_i (6).

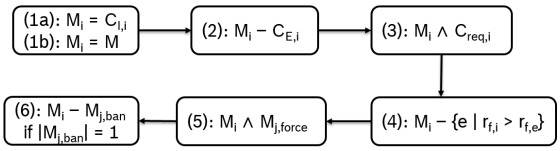

Fig. 4. Reduction sequence

3 CASE STUDY AND RESULTS

Table 1 and Table 2 exemplary show the property values for the optimization problem presented in Figure 1.

	f_1	f_2	f_3	f_4	f_5	f_6
r_f	6	2	3	1	2	3
f_req_f	8	0	4	4	2	1

Table 1. Function requirements

	e_1	e_2	e_3	e_4	e_5
r_e	10	10	5	10	10
f_req_e	8	4	8	2	1

Table 2. ECU properties

original	(1)	(2)	(3)	(4)	(5)	(6)
15,625	9,375	7,500	810	405	243	162
100 %	60 %	48 %	5.2 %	2.6 %	1.6 %	1 %

Table 3. Size of search space after each reduction step

Furthermore, let $C_{I,2} = \{e_1, e_2, e_3\}$, $C_{E,5} = \{e_1\}$, and consider $force(f_5, f_6)$ and $ban(f_1, f_2)$. Applying SSR to the optimization problem results in $M_1 = \{e_1\}$, $M_2 = \{e_2, e_3\}$, $M_3 = \{e_1, e_2, e_3\}$, $M_4 = \{e_1, e_2, e_3\}$, $M_5 = \{e_2, e_3, e_4\}$ and $M_6 = \{e_2, e_3, e_4\}$. As shown in Table 3, the search space could be pruned to about 1 % of its original size. This has also an impact on the number of generations for finding the optimal solutions. With a population size of 10, we commonly find the five Pareto-points after 15 generations, whereas an EA with repair mechanism needs 150 generation and an EA without repair function and mapping sets more than 5000 generations. For a larger use case with 32 functions, 10 ECUs, location constraints on 28 functions, and a population size of 50, Figure 5 shows the mean hypervolume values over 25 optimization runs. Whereas the standalone NSGA-II finds first feasible solutions after 60 generations and then slowly increases, the NSGA-II with SSR converges after 140 generations.

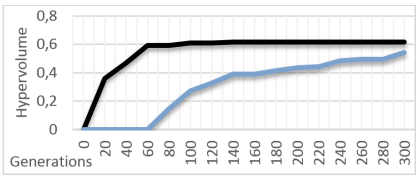

Fig. 5. Hypervolume with SSR (black) and without (blue)

References

1. Blickle, Tobias, Jrgen Teich, and Lothar Thiele. "System-level synthesis using evolutionary algorithms." Design Automation for Embedded Systems 3.1 (1998): 23-58.
2. Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." Evolutionary Computation, IEEE Transactions on 6.2 (2002): 182-197.
3. Hardung, Bernd. Optimisation of the allocation of functions in vehicle networks. Shaker, 2006.
4. Moritz, Ralph, Tamara Ulrich, and Lothar Thiele. "Evolutionary exploration of e/e-architectures in automotive design." Operations Research Proceedings 2011. Springer Berlin Heidelberg, 2012. 361-366.
5. Moser, Irene, and Sanaz Mostaghim. "The automotive deployment problem: A practical application for constrained multiobjective evolutionary optimisation." Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, 2010.
6. Walla, Gregor, et al. "An automotive specific MILP model targeting power-aware function partitioning." Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on. IEEE, 2014