Efficient Constraint Handling during Designing Reliable Automotive Real-Time Systems

Florian Pölzlbauer, Iain Bate, Eugen Brenner

Ada-Europe 2012

- System
  - System Model
  - System Configuration
- Design Constraints
- Constraint Satisfaction
  - Guarantees
  - Constraint Resolving
- Optimization Framework
- Evaluation & Experimental Results
- Conclusion & Outlook

Technology

- hardware-oriented (federated)
- special-purpose hardware-nodes
- software tailored to hardware
- hardware-node executes limited nr. of functionality

Development Process

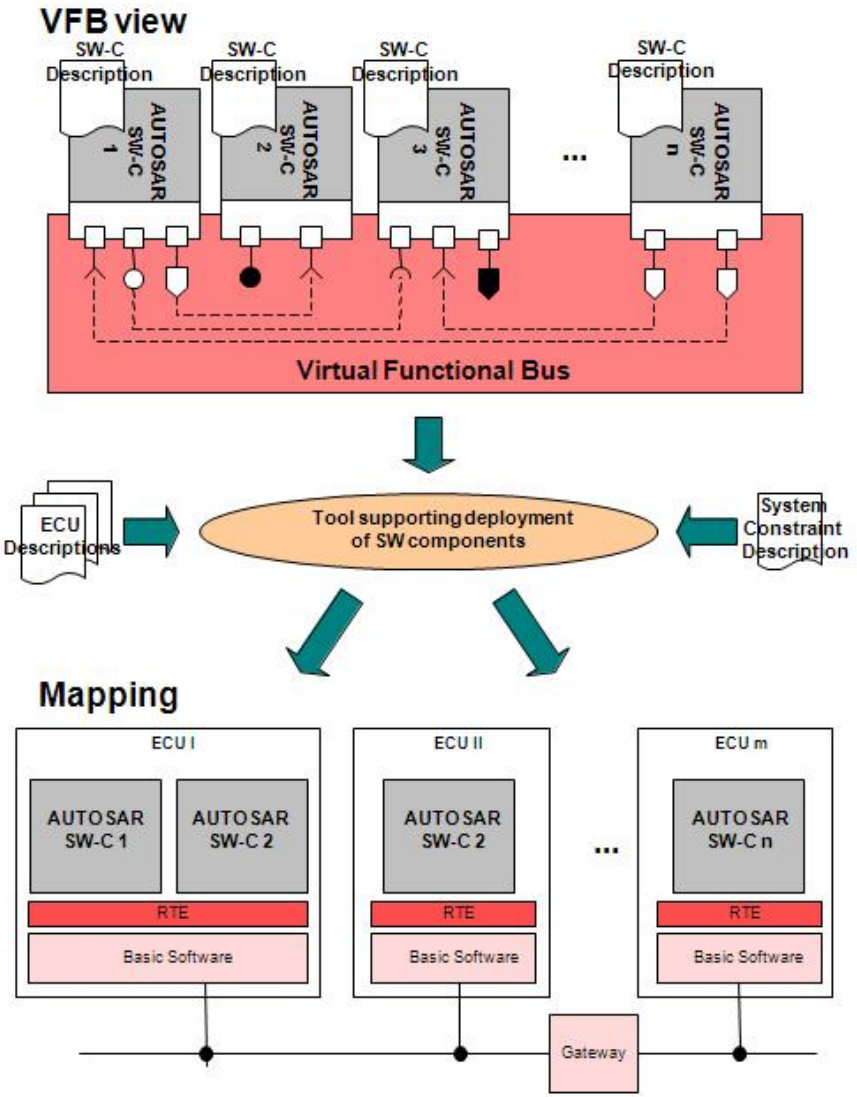
- nodes (hardware incl. software) developed by supplier
- OEM integrates nodes into system
  - integration via specified bus-interfaces

Technology

- software-oriented (software-platform)
- general-purpose hardware
- hardware wrapped by interface-layer / middleware
  - standardized interfaces
  - hardware-details abstracted / hidden
  - software exchangeable between hardware-nodes
  - (e.g. AUTOSAR, IMA, …)

Development Process

- OEMs can buy software-components from supplier
- OEMs have to integrate SW-C into software-platform
  - integration via specified SW-interfaces



© AUTOSAR development partnership

Model

- software architecture
- hardware architecture
  - interface-layer on-top

Configuration Decisions

- software allocation
- data routing
- data-to-frame packing
- scheduling (bus & OS)

- performance analysis

Design Space Exploration

„from scratch“

- all design decisions can be taken
- academic view!

„refinement“

- engineers take upmost important decisions (e.g. safety-related)
- algorithms take remaining decisions
  user decision = constraints for algorithm

„system upgrade“

- system configuration exists
- additional components shall be added
- algorithms find system configuration for extended system
  legacy decisions = constraints for algorithm

Constraint-class	Constraint-type	Literature
A: limited resources	A-1: processor CPU speed	yes
	A-2: processor memory	yes
	A-3: bus bandwidth	yes
B: real-time behaviour	B-1: task deadline	yes
	B-2: communication deadline	yes
	B-3: end-to-end deadline	yes
C: allocation (task to processor)	C-1: dedicated processors	yes
	C-2: excluded processors	yes
	C-3: fixed allocation	yes
D: dependencies (task to task)	D-1: grouping	no*
	D-2: separation	yes
E: data routing (data to bus)	E-1: processor-internal only	no*
	E-2: dedicated buses	no
	E-3: excluded buses	no
	E-4: same bus	no
	E-5: separated buses	no
F: frame packing (data to frame)	F-1: dedicated frame	no
	F-2: same frame	no
	F-3: separated frames	no

Meta-heuristic Search & Optimization

- add cost-term „constraint violations“ → min.
- works for some constraint-types
- inefficient (many violations during search)

Extended Approach

- resolve constraints before search (one time effort)
- add heuristics (for satisfying)
- more efficient (few violations during search)

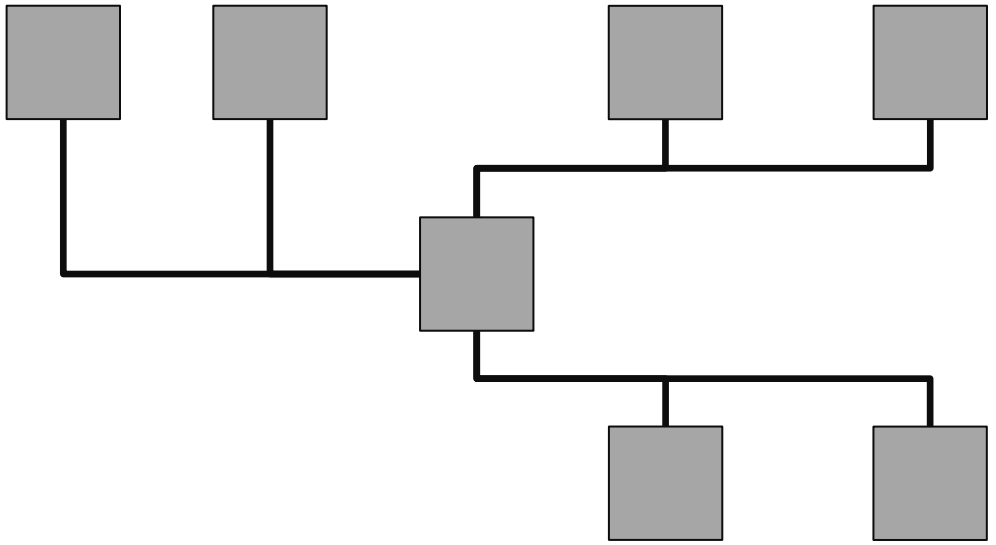
Can we give any guarantee for constraint-satisfaction?

- satisfying & fulfill pre-conditions

- admissible bus-systems

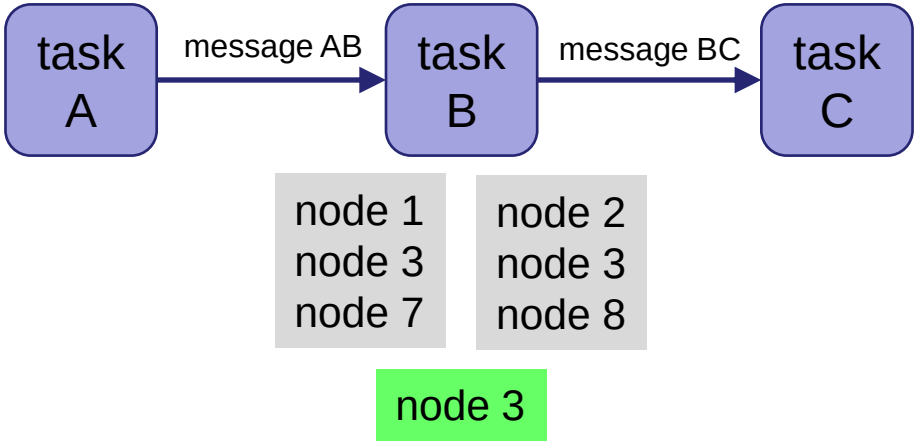
$$B_{adm} = \begin{cases} B \setminus B_{ex} & if \quad B_{ded} = \{\} \\ B_{ded} \setminus B_{ex} & else \end{cases}$$

- initial admissible processors for sender-/receiver-task



- consider all sent/received messages of a task

→ refined set of init. adm. processors



- E-4: routing via same bus
  - group sender tasks
  - group receiver tasks

$$P_{adm} = \begin{cases} (P \cap X) \setminus P_{ex} & if \quad P_{ded} = \{ \\ (P_{ded} \cap X) \setminus P_{ex} & else \end{cases}$$

- D-1: task grouping
  - group tasks (task-cluster)
  - calc set of adm. processors (intersection)

- D-2: task separation
  - dynamically exclude processors

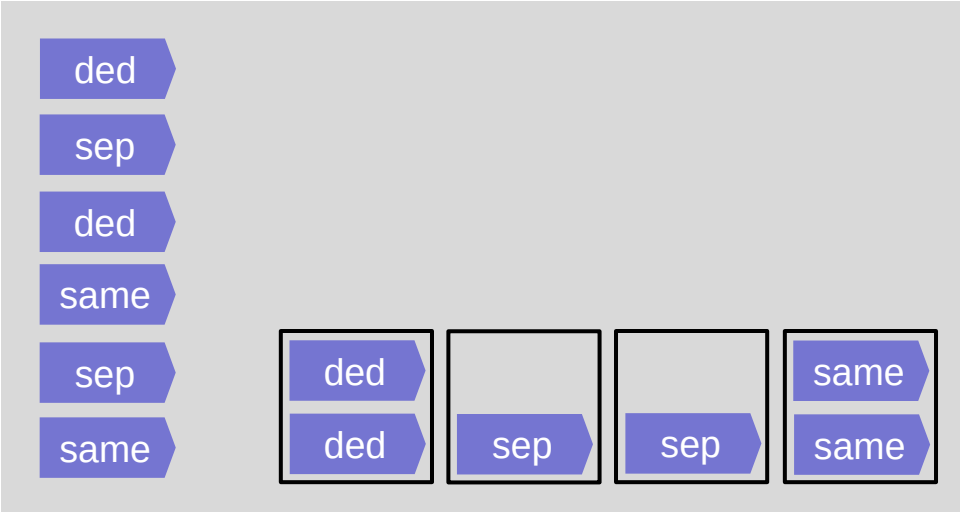
$$P_{adm.dyn} = P_{adm} \setminus P_{ex.dyn}$$

during optimization: only pick from these sets

- perform packing by constraint-aware tailored heuristic
- enforce pre-conditions (data routing)
  - F-2: same frame → same sender & same route

Packing Sequence

1) F-1: dedicated frame
2) F-3: separated frames
3) F-2: same frame
4) unconstrained



Meta-heuristic Search & Optimization

- simulated annealing
- multiple objectives
  - nr. of needed processors → min
  - bus utilization → min
  - processor utilization → balanced
  - nr. of constraint violations → min

$$\cos t = \frac{\sum w_i . c_i}{\sum w_i} \rightarrow \min$$

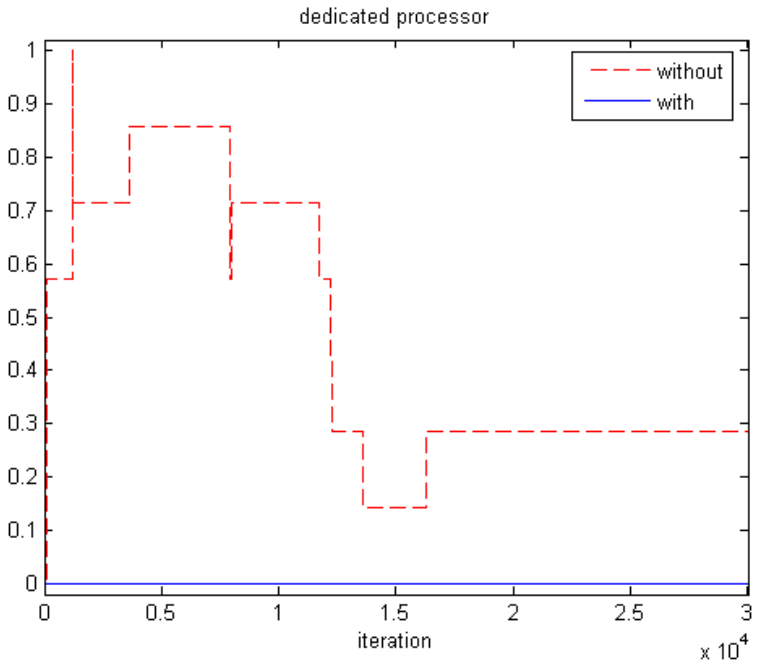
Extensions

- neighbour moves
  - allocation: only pick from admissible processors
  - packing: tailored packing-sequence

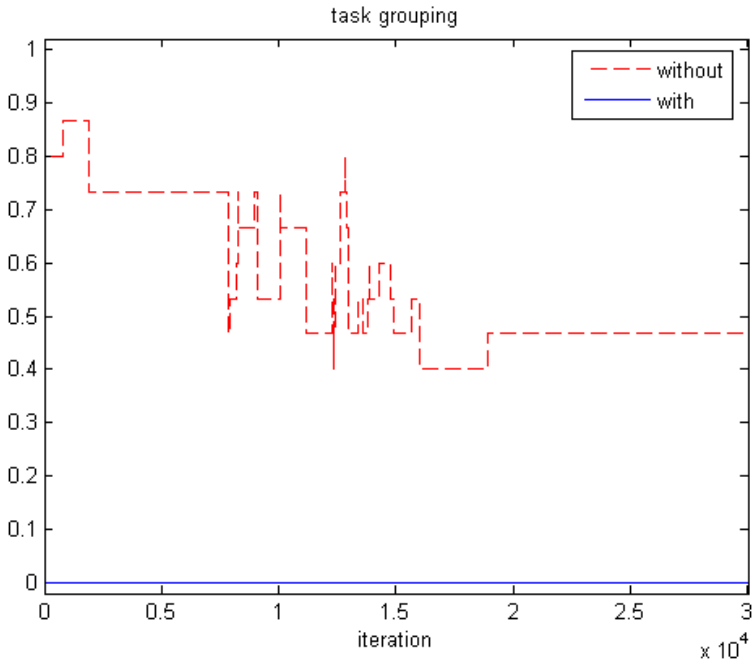
- synthetic problem instances

- “efficient constraint handling”
  - efficient = ?
  - nr. of constraint violations, during search-iterations
  - impact on “best obtained solution”

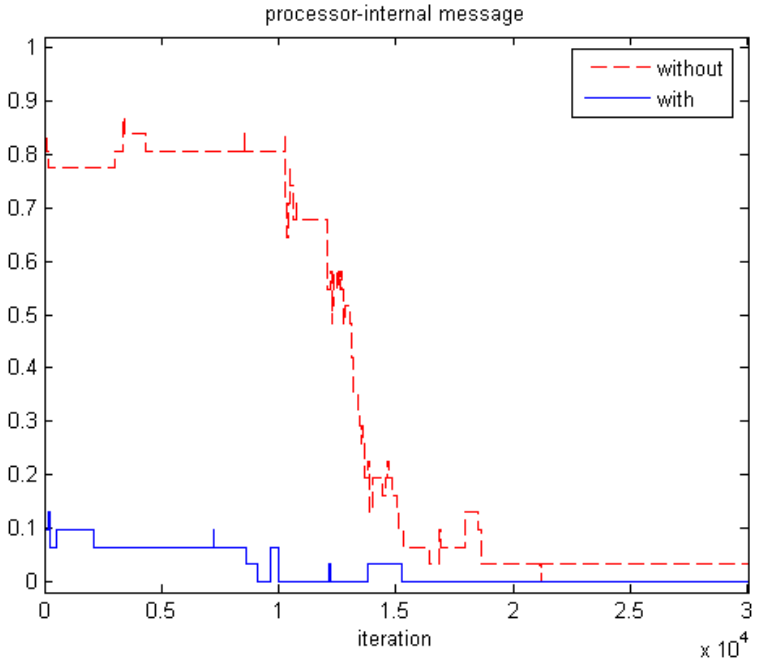
- method: calculate set of admissible resources
- C-1: dedicated processors



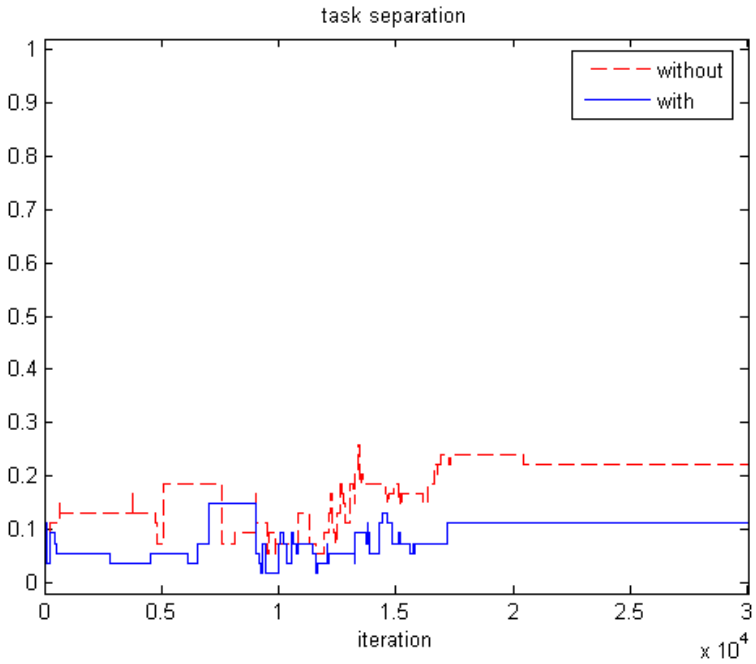
- method: clustering of tasks
- D-1: task grouping



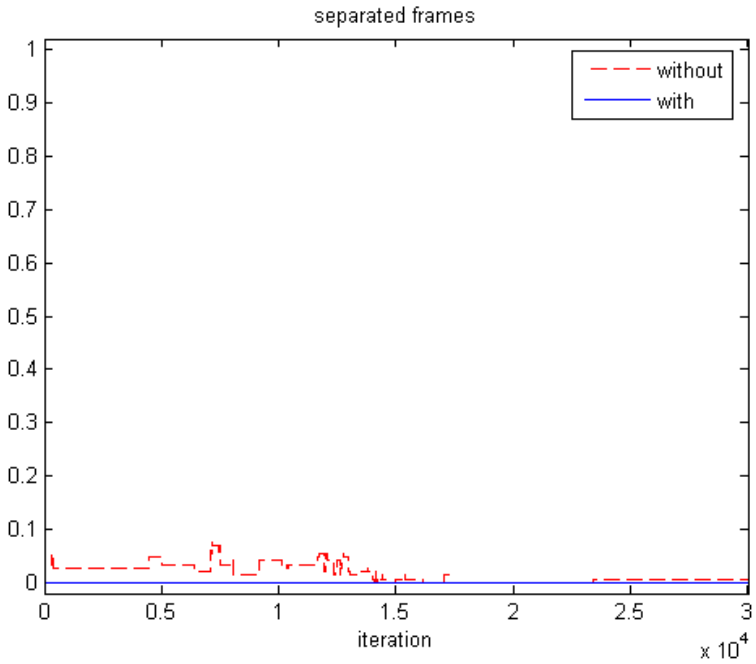
- method: clustering of tasks
- E-1: processor-internal message



- method: dynamically excluding
- D-2: task separation



- method: tailored packing-heuristic
- F-3: separated frames



- F-2: same frame
- no pre-condition enforcing

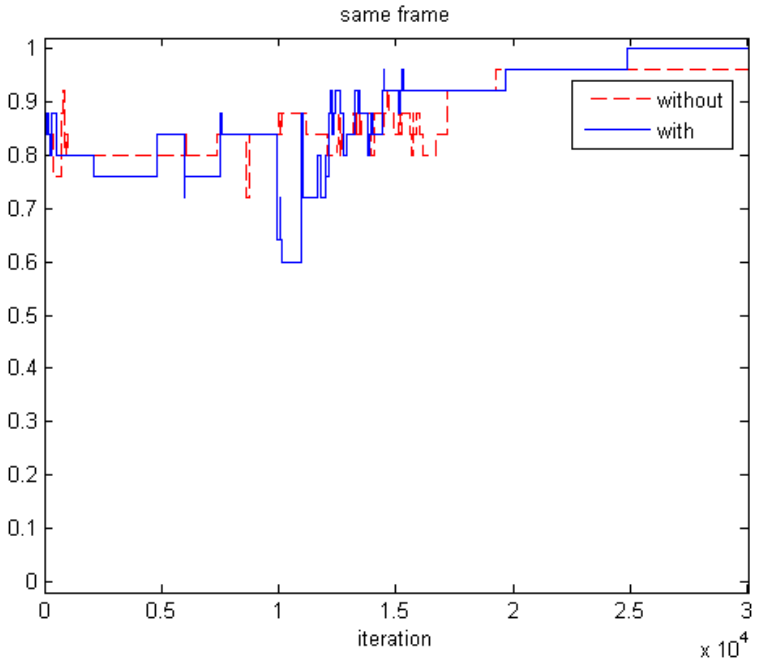


Table 4. Impact of Resolving Constraints on DSE Performance and DSE Results (min./median/max. of 10 runs per scenario)

criteria	no pre-processing	with pre-processing
iterations	10000	10000
unique allocations	9546 / 9588.5 / 9595	9423 / 9430.5 / 9469
feasible allocations	**0 / 1 / 1**	**2026 / 2169 / 2231**
infeasible, due to constr. D-1 & E-1	**9541 / 9578.5 / 9589**	**0 / 0 / 0**
infeasible, due to CPU overload	0 / 1.5 / 8	4050 / 4177 / 4335
infeasible, due to memory overload	1 / 1.5 / 7	2653 / 2693 / 2759
infeasible, due to deadline violation	0 / 2.5 / 6	385 / 406.5 / 421
used processors	5 / 6 / 6 (of 6)	**4 / 4 / 4** (of 6)
bus utilization [%]	12.66 / 13.73 / 14.61	7.52 / 8.23 / 9.87
CPU utilization [%] (average)	47.69 / 47.69 / 57.23	71.54 / 71.54 / 71.54
Δ CPU utilization [%] (average)	7.69 / 10.77 / 20.25	3.85 / 5.19 / 8.08

- constraint resolving improves “best obtained solution”

Conclusion

- objective: find near-optimal system configuration (DSE)
- set of industrial relevant constraint-types
- modular method for constraint-satisfaction
  - resolving & enforce pre-conditions
- increased constraint-satisfaction efficiency
  - fewer violations

→ framework for addressing industrial design scenarios

Outlook

- scheduling constraint-types
  - (e.g. priority range, priority order, …)
- extend approach to multi-core processors

/* */ | ?