Constraint Programming with External Worst-Case Traversal Time Analysis

Constraint Programming 2023

Pierre Talbot, Tingting Hu, Nicolas Navet
pierre.talbot@uni.lu

28th August 2023

University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

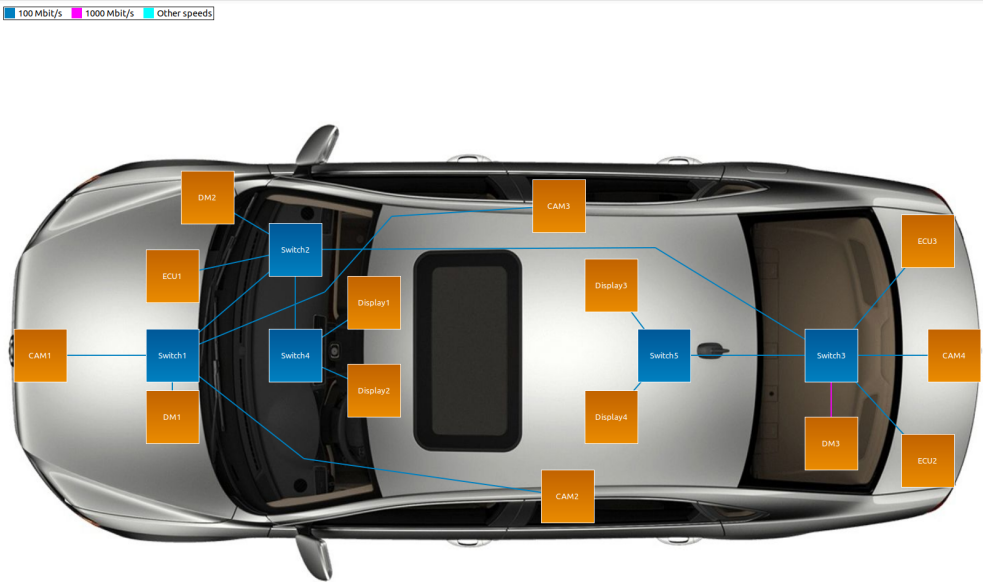
Context of the Work

Collaboration with Nicolas Navet and Tingting Hu.


RTaW
RealTime-at-Work

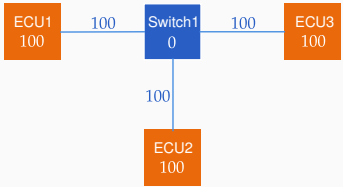
- In the research group *Critical Real-Time Embedded Systems*.
- Connection with the automotive industry (BMW, Mercedes-Benz, Renault, *etc.*)

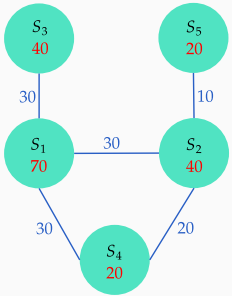
Automotive Network



The Deployment Problem

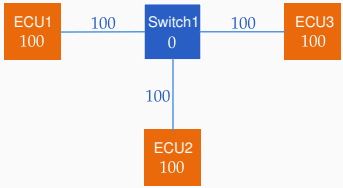
- Given a hardware graph $\langle H, L \rangle$ with 3 ECUs and 1 switch and a software graph $\langle S, Com \rangle$ with 5 services and 5 communications.
- Find a (good) *deployment function* $d : S \to H$ such that all communications among services meet their deadlines.

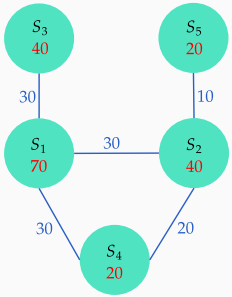




The Deployment Problem

- Given a hardware graph $\langle H, L \rangle$ with 3 ECUs and 1 switch and a software graph $\langle S, Com \rangle$ with 5 services and 5 communications.
- Find a (good) *deployment function* $d : S \to H$ such that all communications among services meet their deadlines.





State of the art (in real-time field)

- **Simulation**: simulate the communications across the network a large number of times with various parameters to show a deployment function works "most of the time".
- **Formal methods (our focus)**: *worst-case traversal time* (WCTT) analysis guarantees the end-to-end delay of network packets, but at the cost of being an incomplete analysis.

Problem: enumerating and analyzing all deployment functions is too time-consuming (about 1.5s per analysis).

Our Approach

- Proposal: Represent WCTT as a constraint problem to prune early unsatisfiable deployment functions.
- $\Rightarrow$ Too difficult: 30 years+ of research in WCTT (network calculus).
- $\Rightarrow$ From the viewpoint of the company: WCTT is already implemented, optimized and working.
- $\Rightarrow$ Represent only the CP-friendly part of the problem and integrate WCTT as an external function called during solving.
- $\Rightarrow$ Extract information from WCTT analysis when it fails and dynamically add it to the model to help searching .

Contributions

- *In distributed embedded system:* A multi-objective constraint model of the deployment problem over Ethernet network.
- *In constraint programming:* A new multi-objective optimization algorithm CUSOLVE_MO integrating external function calls during solving.
- A new use-case of *abstract interpretation* to help understanding the problem, and provide proofs of correctness of the algorithms.

Related Work

Controller Area Network (CAN)

- WCTT over CAN network is simple and actually exact.
- Various approaches to tackle this problem such as with genetic algorithm, MIP, constraint programming.
- Both the constraint part and the analysis part are represented in the model.

Only recently Kugele et al. (2021) considered the deployment over Ethernet networks with SMT solvers and simulation—but small network (3 CPUs), generate-and-test algorithm, and no constraint model provided.

Abstract Interpretation View of the Deployment Problem

Concrete Domain

The *concrete domain* is the set $D^\flat$ of all deployment functions satisfying the deadline constraints. Might be too hard to compute or even uncomputable.

Figure 1: Concrete domain $D^\flat$

Under-approximating Abstract Domain

WCTT analysis under-approximates the concrete domain ($sol(U) \subseteq D^\flat$). It will only accept valid deployment functions, but might reject valid ones.

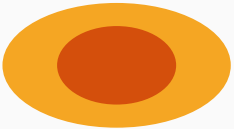


Figure 2: WCTT domain $sol(U)$ (red)

Under-approximating Abstract Domain

WCTT analysis under-approximates the concrete domain ($sol(U) \subseteq D^\flat$). It will only accept valid deployment functions, but might reject valid ones.

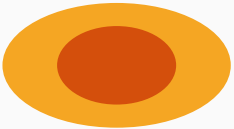


Figure 2: WCTT domain $sol(U)$ (red)

WCTT formally

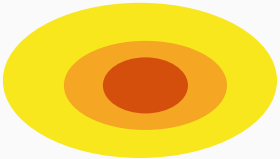
Let $uf : \text{ASN} \to \{true, false\}$ be the external WCTT analysis, then:

$$sol(U) := uf^{-1}(true) = \{asn \in \text{ASN} \mid uf(asn) = true\}$$

Sandwiching the Problem

Use a CP model O over-approximating the concrete domain:

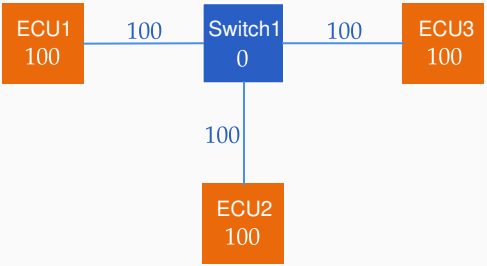
$$sol(U) \subseteq D^\flat \subseteq sol(O)$$

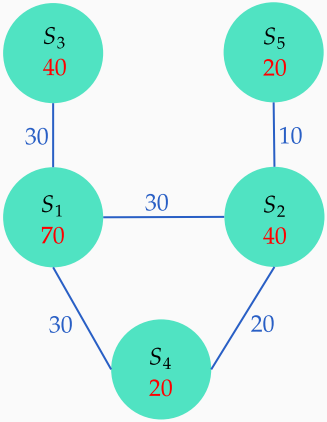


$\Rightarrow$ CP helps pruning unproductive assignments so we call WCTT only on promising assignments.

Over-Approximating Constraint Model

Example


ECU1
100
100
Switch1
0
100
ECU3
100
100
ECU2
100


S_3
40
S_5
20
30
10
S_1
70
30
S_2
40
30
20
S_4
20

Constraint Model of the Deployment Problem

- **Constants**: Set $S = \{s_1, \ldots, s_n\}$ of services and $H = \{h_1, \ldots, h_m\}$ of CPUs.
- **Variables**: $d(s_i) = H$, initially each service s_i can be allocated on any CPU.
- **Constraint 1**: Ensure the utilization rate of each processor (function hc) is not exceeded:

$$\forall h \in H, \quad \sum_{s \in d^{-1}(h)} sc(s) \leq hc(h)$$

  where $sc : S \to \mathbb{Z}$ is the CPU utilization of the services.

- **Constraint 2**: Utilization rate of each network link.

$$\forall \ell \in L, \quad \sum_{c \in Com} com(c, \ell) \leq lc(\ell)$$

  where the function $com(c, \ell)$ returns the cost on the link ℓ of communication c, and is defined by:

$$com(c, \ell) = \begin{cases} cc(c) & \text{iff } \ell \in path(d(x), d(y)),\ c = (x, y) \\ 0 & \text{otherwise} \end{cases}$$

Constraint Model of the Deployment Problem

Multi-objective Optimization Problem

- **Extensibility**: Minimize the maximum utilization rate of a processor:

$$\min \max_{h \in H} \sum_{s \in d^{-1}(h)} sc(s)$$

- **Extensibility**: Same with network link.
- **Cost**: Minimize the number of processors used:

$$\min |d(S)|$$

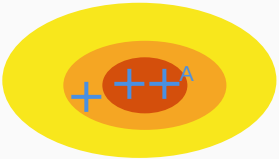
Algorithms

Two Phase Algorithm

Algorithm

1. Compute the Pareto front of the CP model O.
2. Filter the solutions of O passing the WCTT analysis.

But it can prune solutions from $WCTT$, so the filtered Pareto front is not necessarily optimal:



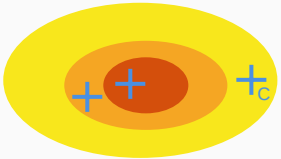
(1a) Compute Pareto front:

Two Phase Algorithm

Algorithm

1. Compute the Pareto front of the CP model O.
2. Filter the solutions of O passing the WCTT analysis.

But it can prune solutions from $WCTT$, so the filtered Pareto front is not necessarily optimal:



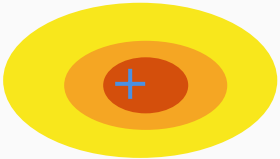
(1b) Compute Pareto front:

Two Phase Algorithm

Algorithm

1. Compute the Pareto front of the CP model O.
2. Filter the solutions of O passing the WCTT analysis.

But it can prune solutions from $WCTT$, so the filtered Pareto front is not necessarily optimal:



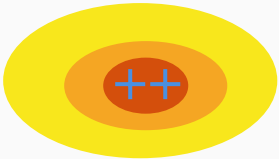
(2) Filtered Pareto front:

Two Phase Algorithm

Algorithm

1. Compute the Pareto front of the CP model O.
2. Filter the solutions of O passing the WCTT analysis.

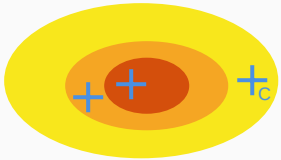
But it can prune solutions from $WCTT$, so the filtered Pareto front is not necessarily optimal:



What we wanted to obtain:

Two Phase Algorithm Fixed?

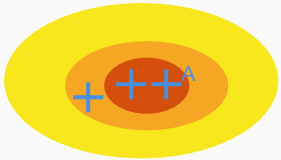
We can keep the intermediate solutions generated, then when filtering, if any solutions is discarded by WCTT, we reconstruct the previous Pareto front without this solution.



Filtering Pareto front:

Two Phase Algorithm Fixed?

We can keep the intermediate solutions generated, then when filtering, if any solutions is discarded by WCTT, we reconstruct the previous Pareto front without this solution.

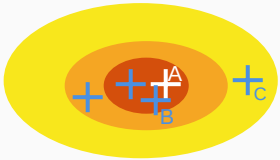


Filtering Pareto front:

Two Phase Algorithm Fixed?

We can keep the intermediate solutions generated, then when filtering, if any solutions is discarded by WCTT, we reconstruct the previous Pareto front without this solution.

Can still discard optimal solutions (suppose point B dominates A but is dominated by C):



Therefore, we must call WCTT during solving for completeness.

Integrated Algorithm: usolve_mo

function USOLVE_MO(O, uf, $\sqcup$, opt)
  $F \leftarrow \{\}$
  $asn \leftarrow \text{SOLVE}(O)$
  while $asn \neq \{\}$ **do**
    if $uf(asn) = \texttt{true}$ **then**
      $F \leftarrow F \sqcup \{asn\}$
      $O \leftarrow O \wedge opt(asn)$
    else
      $O \leftarrow O \wedge \neg asn$
    end if
    $asn \leftarrow \text{SOLVE}(O)$
  end while
  return F
end function

$\Rightarrow$ Proposition. The Pareto front computed is optimal w.r.t. $sol(U)$.

Conflicts Generated by WCTT

Conflicts from *uf*

The WCTT analysis returns which communications fail to pass the analysis.

$\Rightarrow$ Generate new constraints based on this information.

$\Rightarrow$ $uf : \text{ASN} \rightarrow C$.

$\Rightarrow$ Requirement: if $uf(asn) \neq true$, then $uf(asn) \Rightarrow \neg asn$.

Examples of Conflicts

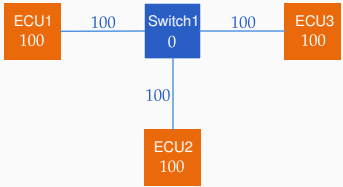
Suppose the communication between the services x and y fail:

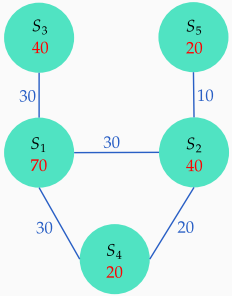
- Forbid source (FS): $d(x) \notin \{asn(x), asn(y)\}$
- Forbid target (FT): $d(y) \notin \{asn(x), asn(y)\}$
- Decreasing hops (DH): $|path(d(x), d(y))| < |path(asn(x), asn(y))|$

Conflicts are Generally not Over-approximating

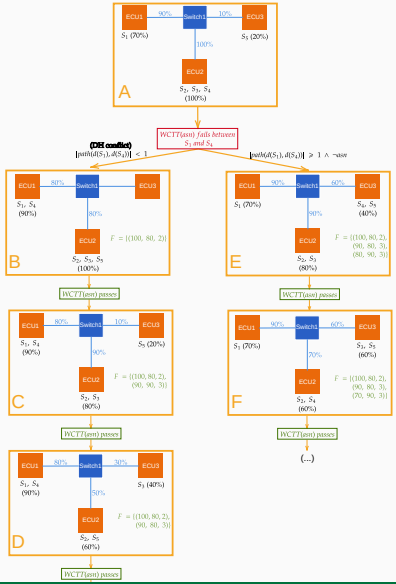
None of the conflicts presented are over-approximating, i.e., adding them to O might prune some solutions. We propose the algorithm CUSOLVE_MO to use conflicts as heuristics without losing completeness.

Example:





cusolve_mo: example



cusolve_mo

function CUSOLVE_MO(F, O, C, uf, $\sqcup$, opt)
  $asn \leftarrow$ OSOLVE($O \wedge C$)
  if $asn \neq \{\}$ **then**
    $co \leftarrow uf(asn)$
    if $co =$ `true` **then**
      $F \leftarrow F \sqcup \{asn\}$
      $O \leftarrow O \wedge opt(asn)$
      CUSOLVE_MO(F, O, C, uf, $\sqcup$, opt)
    else
      CUSOLVE_MO(F, O, $C \wedge co$, uf, $\sqcup$, opt)
      CUSOLVE_MO(F, O, $C \wedge \neg co \wedge \neg asn$, uf, $\sqcup$, opt)
    end if
  end if
  return F
end function

$\Rightarrow$ Proposition. The Pareto front computed is optimal w.r.t. $sol(U)$.

Experiments

Description of the Experiments

Setting

- AMD Epyc ROME 7H12 processor (64 cores, 280W).
- SOLVE implemented by GECODE 6.3.0 in parallel mode with 8 cores (16 threads), timeout 30 minutes.

Experiments

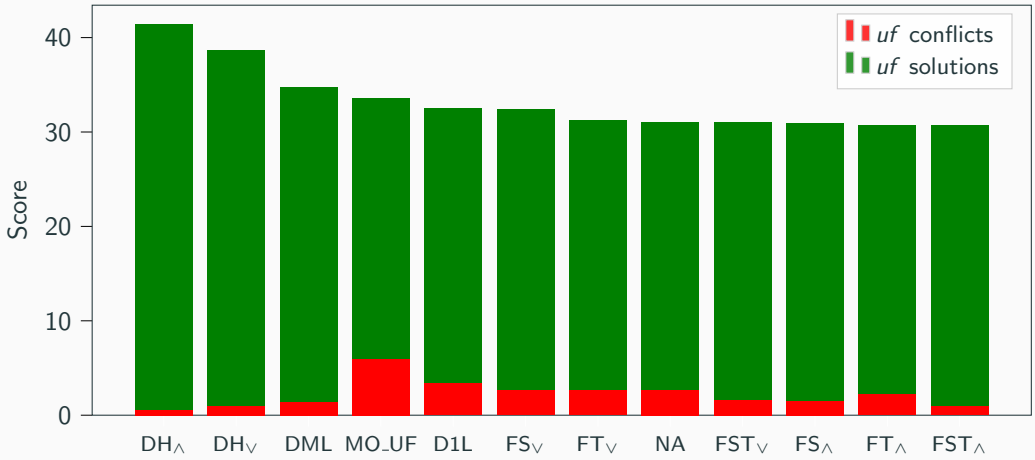
- Instances derived from a realistic automotive Ethernet network consisting of 19 network devices (14 ECUs and 5 switches).
- 5 instances of 50 services, 5 instances of 75 services and 8 instances of 100 services.
- For each of the 18 instances, we generated 10 versions where the sum of all computational requirements is 20%, 40%, 60%, 80% and 90% of the total computational capacity of all ECUs with a uniform distribution among services.

Two Phase Algorithm

- Low number of services (50): 14/15 instances have the same hypervolume before and after filtering.
- For 75 and 100 services: 24/30 instances with an hypervolume within 3% of the unfiltered hypervolume.
- Still, some instances with filtered hypervolume below 75% of the unfiltered hypervolume.

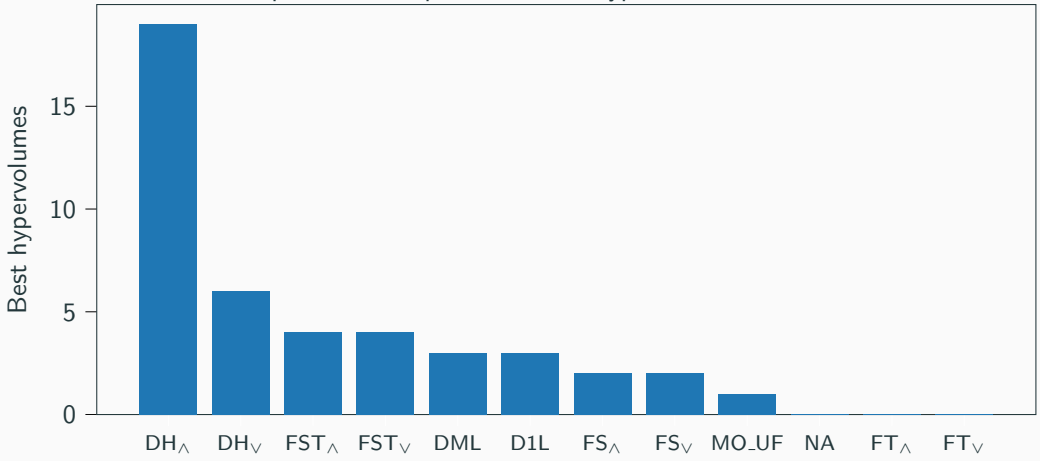
Cumulated Hypervolume Score

Cumulated hypervolume score for each experiment over all instances.



Best Hypervolume

Number of times each experiment computed the best hypervolume.



Conclusion

- Problems from the industry are often unpure.
- Need to reuse existing code, external blackbox functions.
- Rigorous and generic algorithm for CP + under-approximating external function.

Ad: Recruiting a Postdoc

GPU + CP + Abstract Interpretation

(At the university of Luxembourg, 80k€ gross salary, 16 months with extension possible).