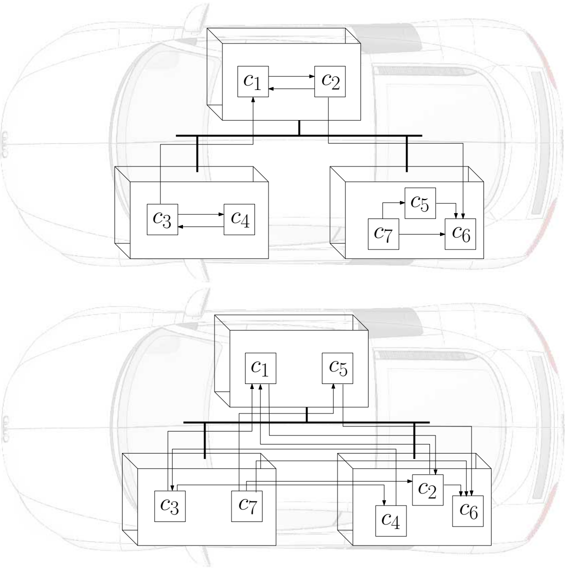
Adaptive Neighbourhood Search

for the Component Deployment Problem

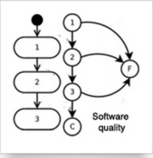
Aldeida Aleti & **Madalina M. Drugan**

October 27, 2015

Software architecture design



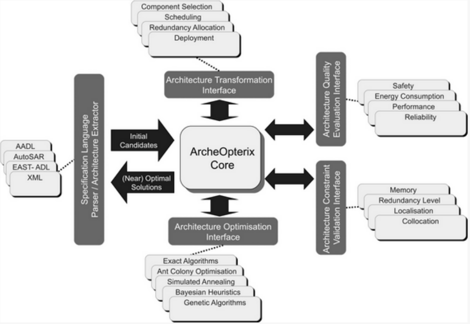
Software quality optimisation



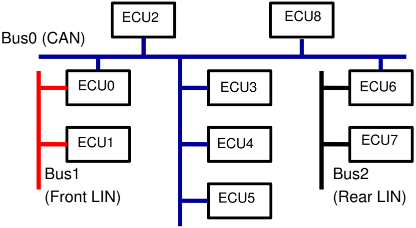
The design of embedded systems involves several important decisions, such as choosing the software components to use, and deciding how to deploy them into the hardware platform. These decisions affect the quality attributes of the system, such as reliability and safety. Embedded systems are becoming more complex with many design options to choose from. We have automated this task.

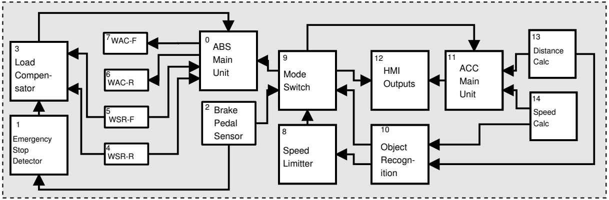
ArcheOpterix

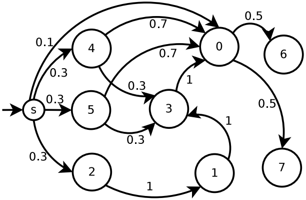
ArcheOpterix is a generic platform for modelling, evaluating and optimising embedded systems. The main modules of ArcheOpterix are shown in the figure below.

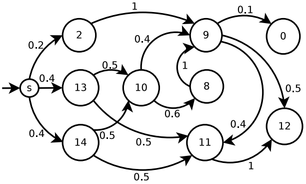


Component Deployment Problem



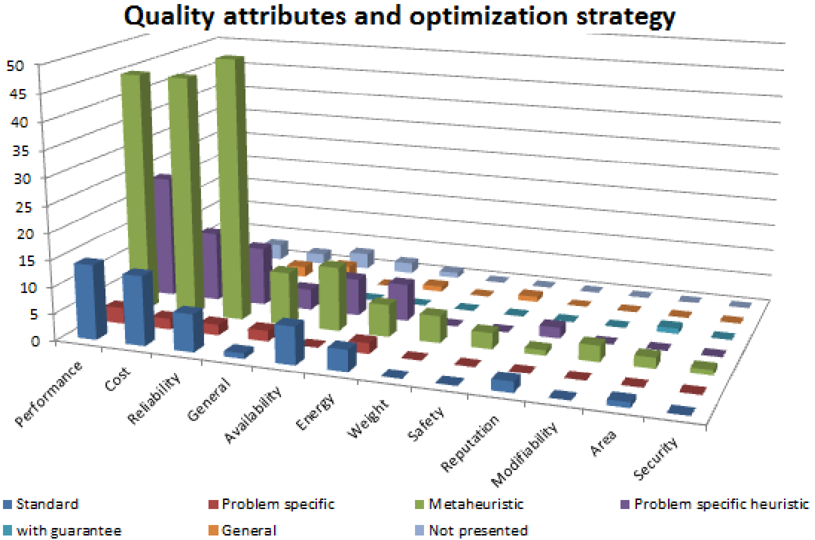






WAC : Wheel Actuator Controllers (Front and Rear)
WSR : Wheel Sensor Readers (Front and Rear)

Problem Difficulty



Variables in Component Deployment optimization

Software components $\mathbb{C} = \{c_1, c_2, \ldots, c_n\}$ (discrete fix number)

- memory size sz_i of the i-th component (KiloBytes)
- workload $wl_i \rightarrow$ million instructions
- initialization probability q_i to start from the i-th component

Interaction between software components

- data size $ds_{ij} \rightarrow$ the amount of data sent between c_i and c_j in a single communication event
- execution probability $p_{ij} \rightarrow$ the probability that calling c_i ends with a call of c_j

Hardware hosts $\mathbb{H} = \{h_1, h_2, \ldots, h_m\}$ (discrete fix number)

- memory capacity cp of each host (KiloBytes)
- processing speed $ps \rightarrow$ instruction - process capacity of hardware unit
- failure rate $fr \rightarrow$ the probability of a single hardware failure

Hardware links $\mathbb{N} = \{n_1, n_2, \ldots, n_s\}$

- data rate $dr_{ij} \rightarrow$ data transmission rate of a bus
- failure rate $fr_{ij} \rightarrow$ data communication error at each link

Component Deployment optimization

$D = \{d \mid d : \mathbb{C} \to \mathbb{H}\} \to$ the set of all function assignment components to hardware resources

Objective function

- *reliability of a component* i is $R_i = e^{-\mathrm{fr}_{d(c_i)} \cdot \frac{\mathrm{wl}_i}{\mathrm{ps}_{d(c_i)}}}$
- *reliability of communication* between components c_i and c_j

$$R_{ij} = e^{-\mathrm{fr}_{d(c_i)d(c_j)} \cdot \frac{\mathrm{ds}_{ij}}{\mathrm{dr}_{d(c_i)d(c_j)}}}$$

- *expected number of visits for each component* $v : C \to \mathbb{R}_{\geq 0}$

$$v_i = q_i + \sum_{j \in \mathcal{I}} v_j \cdot p_{ji}$$

- *expected number of visits of network links* $v : C \times C \to \mathbb{R}_{\geq 0}$

$$v_{ij} = v_i \cdot p_{ij}$$

- *reliability of a deployment architecture* $d \in D$ is $R = \prod_{i=1}^{n} R_i^{v_i} \prod_{i,j} R_{ij}^{v_{ij}}$

Representation and Pertubator operators

Solution representation $\leftarrow$ **string of (component,host) pairs**

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
h_1	h_3	h_3	h_2	h_4	h_1	h_2	h_4

OneFlip operator

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
h_1	h_3	h_3	h_2	h_4	h_1	h_2	h_4

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
h_1	h_3	h_4	h_2	h_4	h_1	h_2	h_4

kOpt operator, where $k = 2$

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
h_1	h_3	h_3	h_2	h_4	h_1	h_2	h_4

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
h_1	h_3	h_1	h_2	h_4	h_3	h_2	h_4

Perturb $\leftarrow$ **kOpt with random value for** $k \in \{1, \ldots, n\}$

Adaptive Neighbourhood Search

```
procedure AN
    S = RandomlyAllocate(C, H)
    N = SelectNeighbourhoodOperator(P(N))
    if N == OneFlip then
        Q(N)=OneFlip(S)
    end if
    if N == KOpt then
        Q(N)=KOpt(S, k)
    end if
    if N == Perturb then
        Q(N)=Perturb(S)
    end if
    ReportFeedback(Q(N))
    Return(S)
end procedure
```

The selection of the neighbourhood operator is a fitness proportionate method $P(N) = \alpha P(N) + (1 - \alpha)Q(N)$.

Neighbourhood operators for the Component Deployment

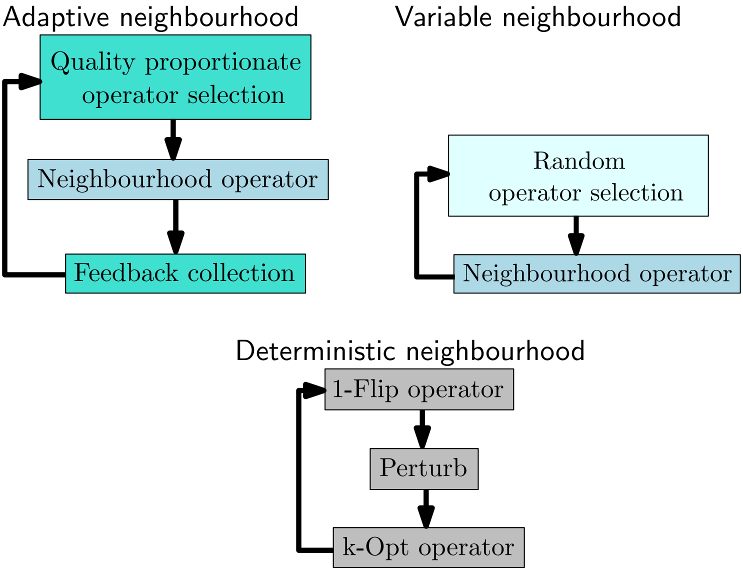
procedure OneFlip(S)
    $S^* = S$
    localOptimum = True
    for all $c < C$ **do**
        h = RandomlySelectHost(H)
        S' = AssignComponentToHost(S, c, h)
        Evaluate(S')
        if $S' > S^*$ **then**
            $S^* = S'$
        end if
    end for
    improvement = FitnessDifference(S, S^*)
    $S = S^*$
    Return(improvement)
end procedure

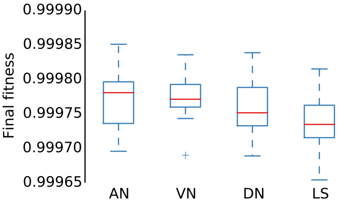
Deterministic neighbourhood search

```
procedure DN
    S = RandomlyAllocate(C, H)
    OneFlip(S)
    Perturb(S)
    KOpt(S, k)
    Return(S)
end procedure
```

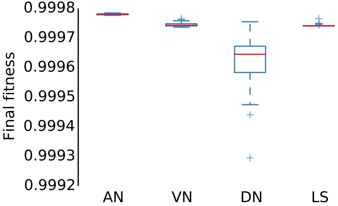
Variable neighbourhood search

```
procedure VN
    S = RandomlyAllocate(C, H)
    r = Random([0,1])
    if r > 0.5 then
        OneFlip(S)
    end if
    if r < 0.5 then
        KOpt(S, k)
    end if
    p = Random([0,1])
    if p < 0.01 then
        Perturb(S)
    end if
    Return(S)
end procedure
```

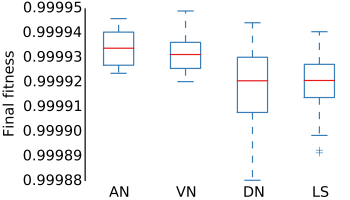

Adaptive neighbourhood
Quality proportionate operator selection
Neighbourhood operator
Feedback collection
Variable neighbourhood
Random operator selection
Neighbourhood operator
Deterministic neighbourhood
1-Flip operator
Perturb
k-Opt operator



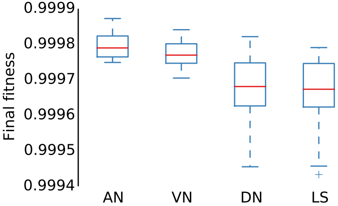
(a) H30C65



(b) H42C85



(c) H55C107

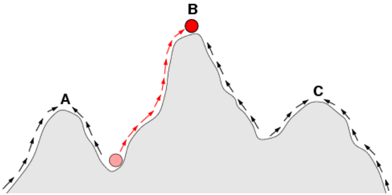


(d) H62C130

Figure: 30 trials (KS-test p-value<0.05).

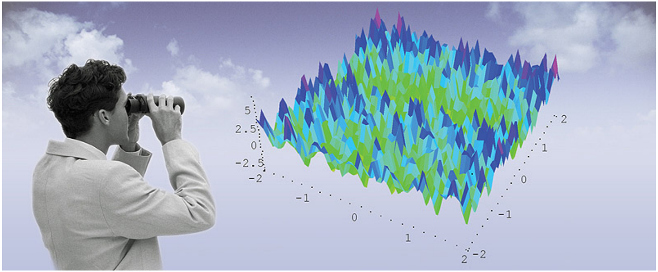
Alternative optimization methods

- Genetic algorithms
- Simulated annealing
- Steepest descent
- Ant colony optimisation
- Hill-climbing
- and the list goes on



Remarks

- The nature of the search-space is the key factor determining the performance of the optimisation algorithm,
- Define/characterise the search-space,
- Analyse what makes problems difficult,
- Guide the optimisation process
  - Select the right search strategy



Future work: difficulty of local search