

A General Framework for Combined Module- and Scale-based Product Platform Design

L. Liberti¹, O. De Weck², D. Krob¹, F. Marinelli³, A. Mucherino¹

¹ LIX, École Polytechnique, France

² ESD, MIT

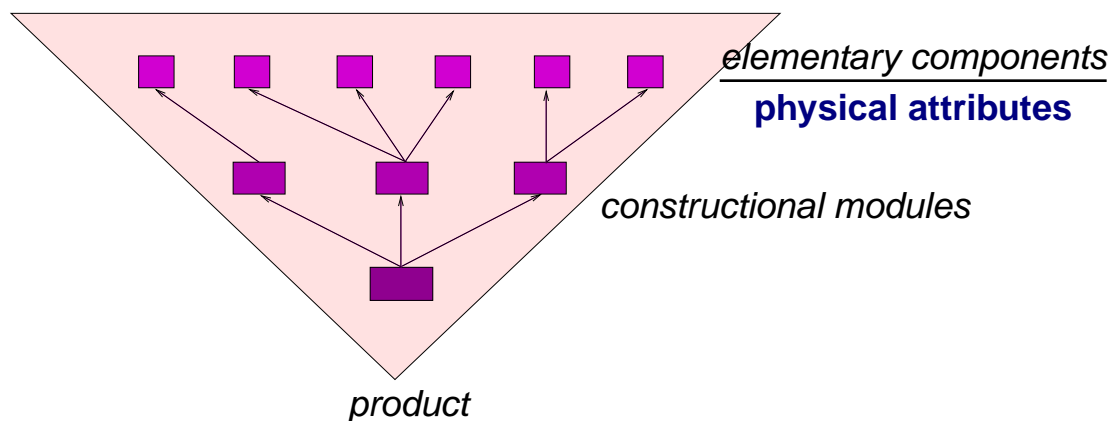
³ DIIGA, Università Politecnica delle Marche, Italy

The overall approach

- **Problem:** determine a *product platform* so as to maximize commonality (or optimize a cost function)
- **Approach:** formalization of the problem by means of *mathematical programming* (MP)
- **Solution:** via Branch-and-Bound by an off-the-shelf solver interfaced with an MP language environment

Products: PBS

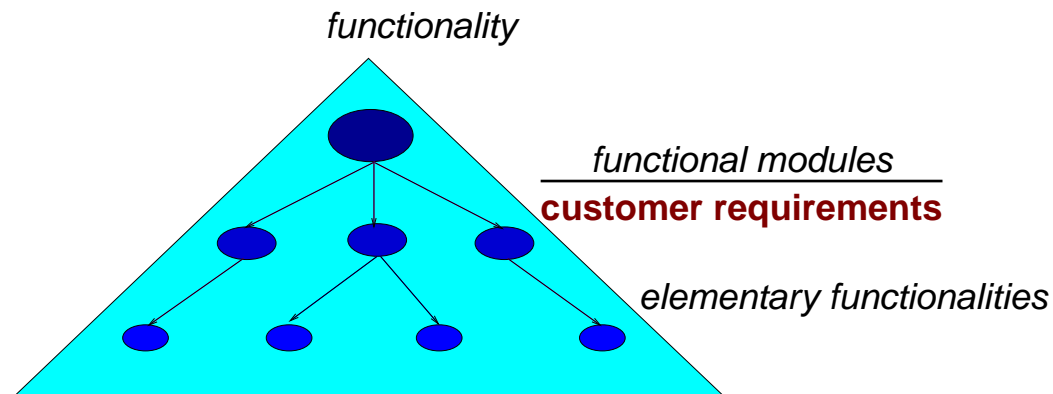
- A *product* p is a set of physical components u together with a physical architecture specifying how the components are put together
- The physical architecture is given as a tree $PBS(p)$ (the Physical Breakdown Structure) whose leaves are the **physical elementary components**



- To each elementary component u we associate a set of design bounds on its **physical attributes**, e.g. *weight*, *colour*,...

Products: FBS

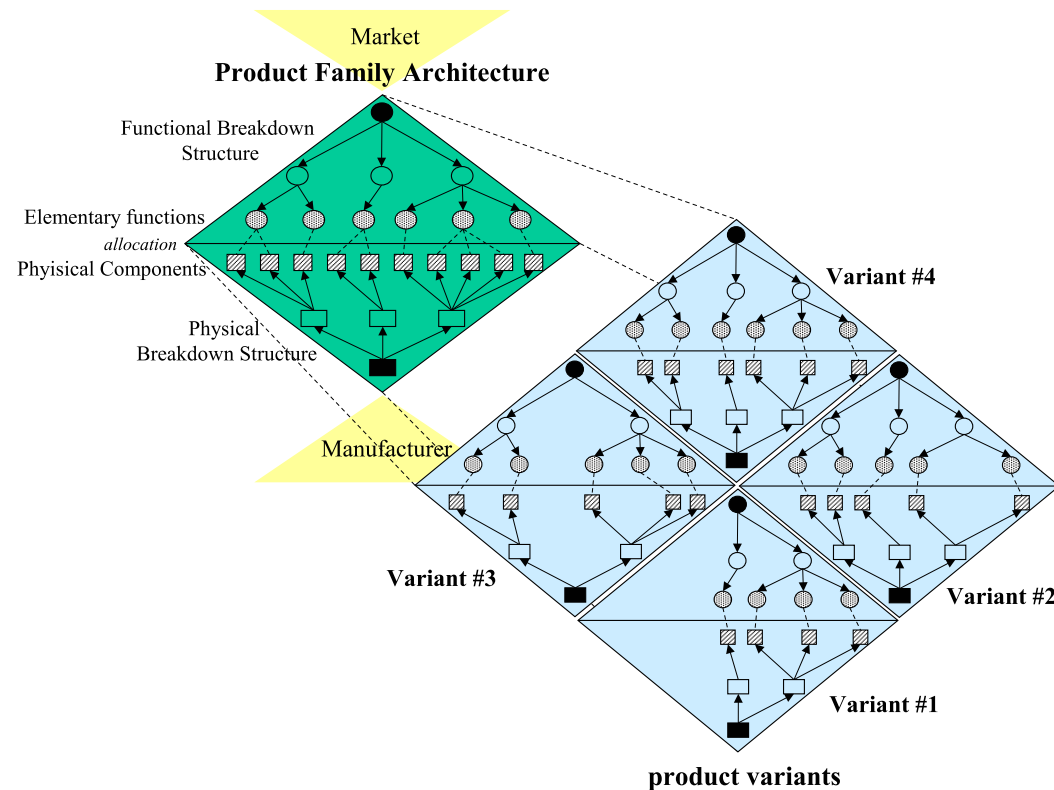
- A product may serve several purposes. The *functionality* of a product i is the set of all purposes it serves.
- The functionality is described by a tree $FBS(i)$ (the Functional Breakdown Structure) which details the hierarchical relations among the purposes



- Each vertex of $FBS(i)$ is called a *functional module*
- The leaves of $FBS(i)$ are the *elementary functionalities*
- To each functional module v of i , we associate a set of **customer requirements**, e.g. *operating system, optional features, weight, ...*

Product families

- A *variant* of a product i is a product j such that there are “few” differences between $PBS(i)$ and $PBS(j)$
- A *product family* P is a set of product variants
- Product families address different market needs with few changes to the PBS of an existing product



The laptop family example



Variants (top-level FM):

basic, professional, traveller

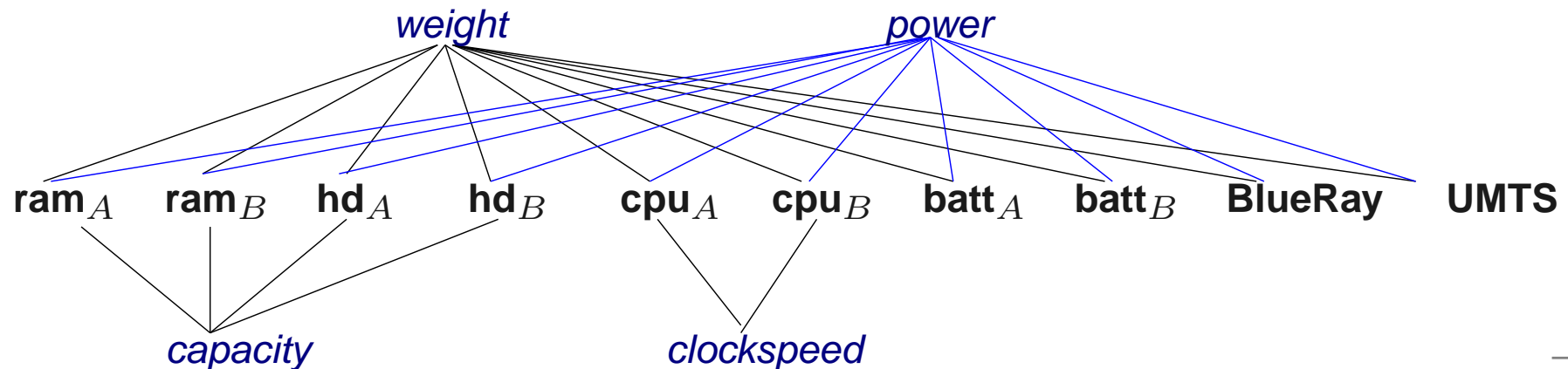
Customer requirements:

weight, operating system, connectivity

Physical attributes:

weight, power, capacity, clockspeed

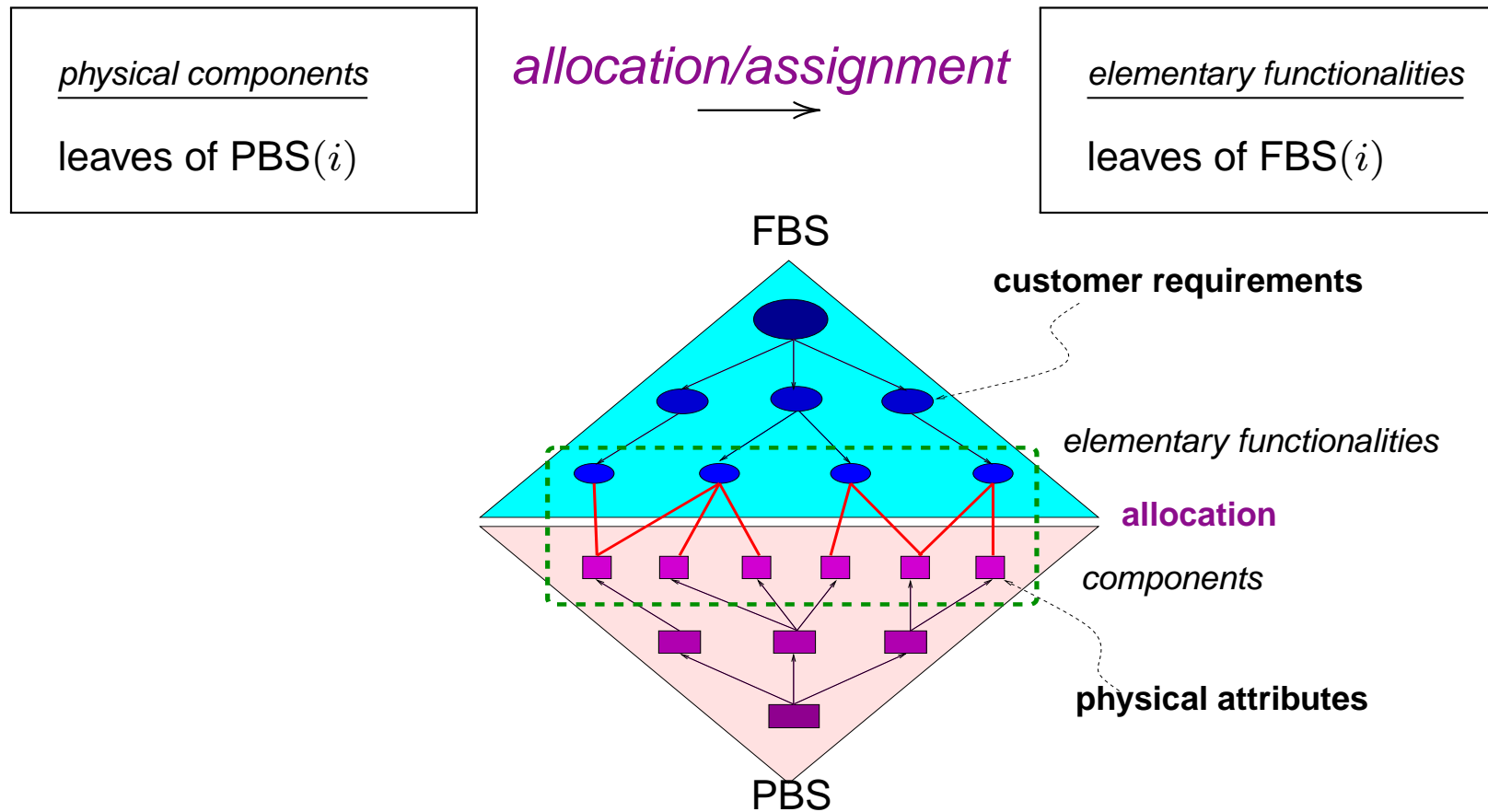
<i>Cust. req.</i> Variants	<i>Wt (kg)</i>	<i>OS</i>	<i>Conn</i>
Basic	[0.8,1.5]	{Xp}	NO
Professional	[1.0,2.0]	{Xp, Vista, Linux}	YES
Traveller	[0.5,1.0]	{Xp, Vista}	YES



Platforms

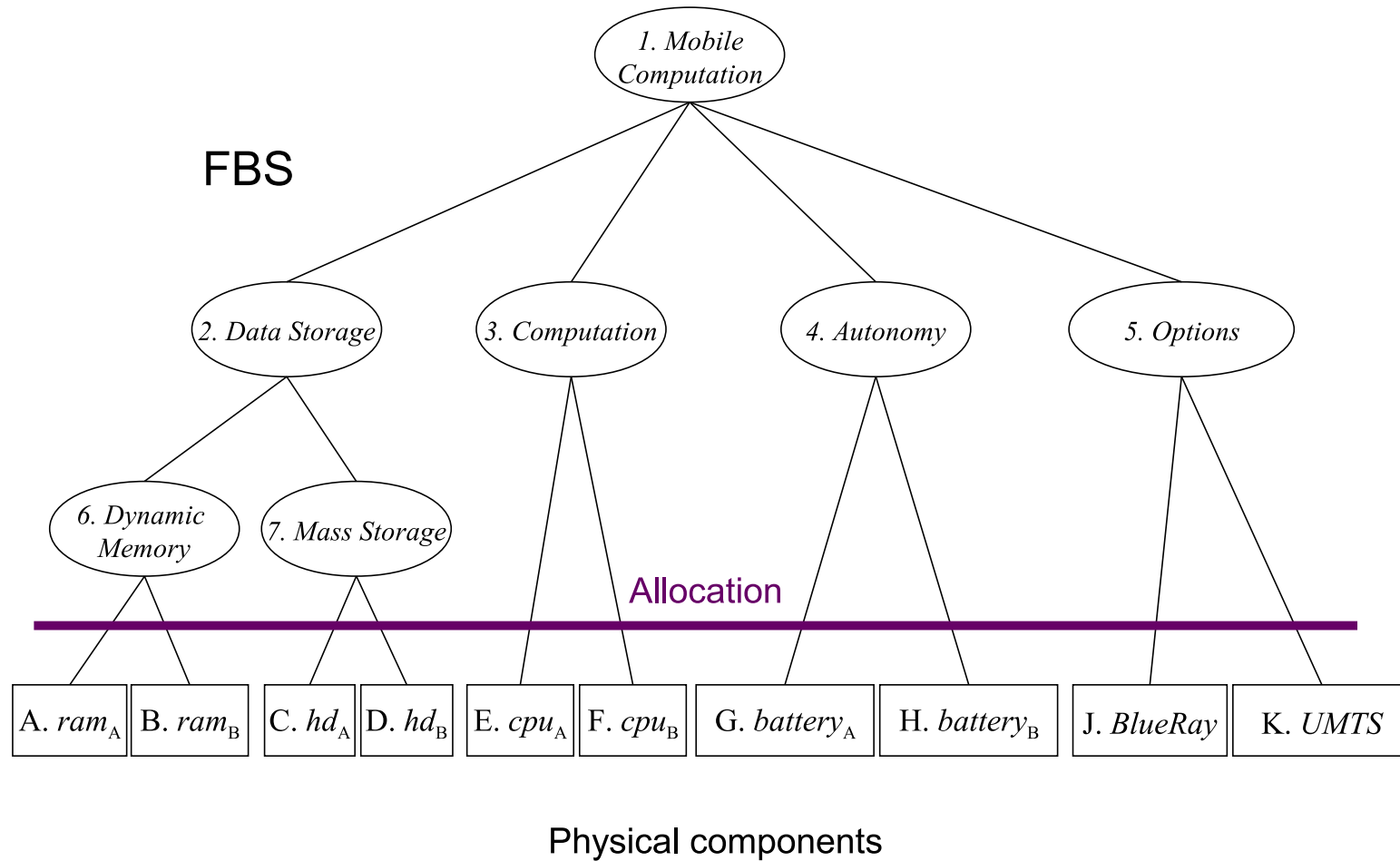
- A **platform** is the collection of assets, i.e, components processes, knowledge, people and relationships that are shared by the products of a family
- A platform P is *feasible* if:
 - for each $i \in P$, $FBS(i)$ satisfies the given set of **customer requirements** on the functional modules
 - for each $i \in P$, the **physical attributes of the elementary components** $PBS(i)$ satisfy the given set of **design bounds** on the physical components

Optimal platforms



A feasible platform is *optimal* if this **allocation decision** occurs in such a way as to minimize the PBS differences over all the variants in the platform

Laptop FBS



Decision?

- PROBLEM INPUT (given parameters):
 - List of elementary components with their design bounds
 - all variants' FBS topologies with per-node customer requirements
- PROBLEM OUTPUT (decisions):
 - allocation of elementary components to elementary functionalities
 - values for physical attributes satisfying customer requirements and design bounds

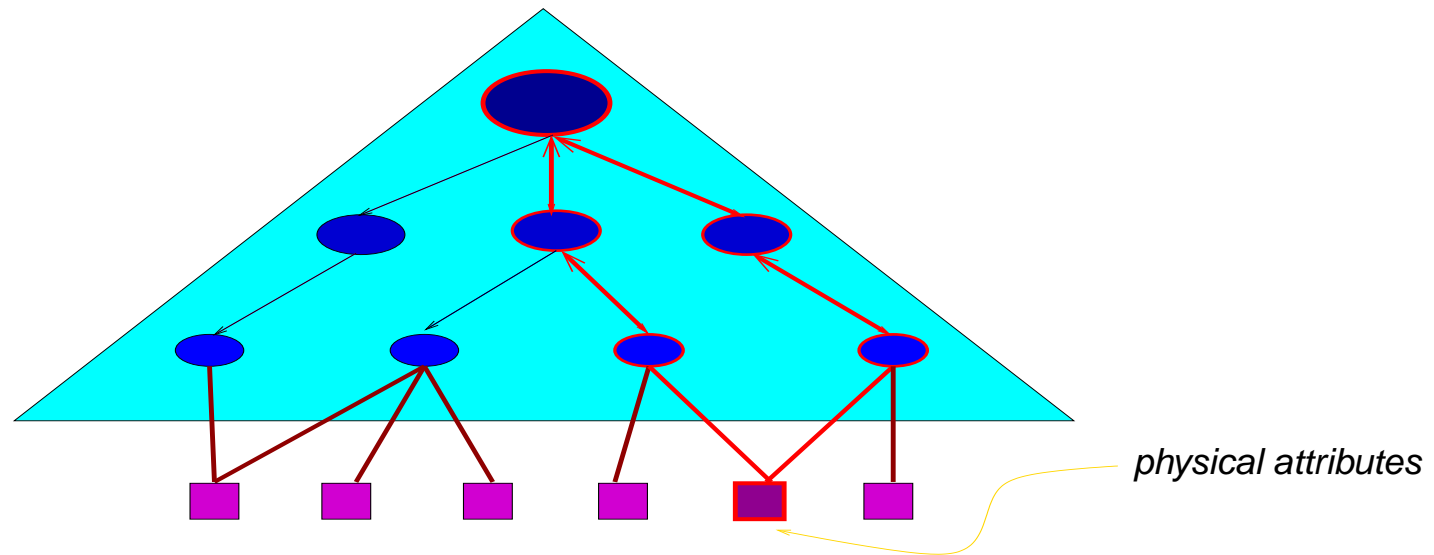
Troubles:

- (1) *physical attributes* are in PBS, *customer requirements* are in FBS
- (2) *they may be incomparable quantities*

Attribute propagation

- Physical attributes: properties of physical components; via the **allocation**, also of elementary functionalities
- Propagated upwards to the parent module via the *composition function* δ (submodule **attributes** to module **attributes**)

$$\text{Example. } \delta(\text{weight}_v) = \sum_{u \in \text{submodules}(v)} \text{weight}_u$$



⇒ Every functional module has associated attributes

Mapping attributes to requirements

- **Customer requirements**: properties of functional modules (vertices of $FBS(i)$)
- The **attributes** of each functional module are mapped into **customer requirements** via the *requirement function* ψ

Example. $\psi(\textit{shipping}, \textit{material}, \textit{color}) = \textit{cost}$

Compatibility

- Compatibility of two functional modules in the same product variant: given by a *module interface function* φ

Example. $\varphi(\text{laptop}, \text{desktop}) = \text{false}$

- Compatibility of two physical components in the same product variant: given by a *component interface function* χ

Example. $\chi(\text{bus_USB1.1}, \text{disk_USB2}) = \text{false}$

The model

- \forall product p , for all module $\in \text{FBS}(p)$, the **functional attributes** must satisfy the **customer requirements**
- Get constraints on **attribute values** and **allocation decisions**:

mapping (**propagation** (**attributes** (**allocation**)))
 \in **customer_requirements**

- A suitable *objective function* minimizes the number (or cost) of different physical components (part numbers) used across the platform
- This problem can be modelled by a *mathematical program* and solved with existing off-the-shelf software

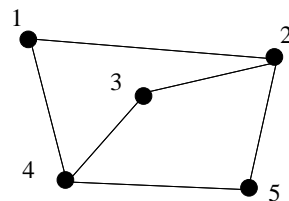
Mathematical Programming

- Mathematical Programming (MP) is a *formal language* for describing optimization problems
- Each MP is a 5-tuple (S, I, X, O, C) :
 - S is a collection of index sets
 - I is a set of parameters (S, I : problem input)
 - X is a set of decision variables (the problem output)
 - O is a set of objective functions f mapping I, X to \mathbb{R} and an optimization direction in $\{\min, \max\}$
 - C is a set of constraints $L \leq g \leq U$ where $L, U \in \mathbb{R}^m$ and g maps I, X in \mathbb{R}^m
 - the symbols in I, X, O, C are indexed on sets in S

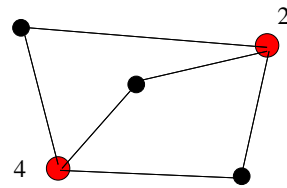
MP example

Problem

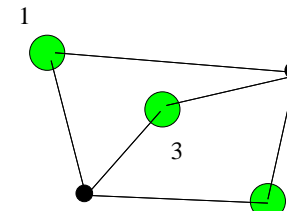
WEIGHTED MAXIMUM INDEPENDENT SET. Given an undirected graph $G = (V, E)$ with node weights w find the subset $U \subseteq V$ such that $w(U)$ is maximum and $\forall u, v \in U (\{u, v\} \notin E)$



$G = (V, E)$



ind. set of size 2



ind. set of size 3

Mathematical program

INDEX SETS	S	V, E	vertex, edges
PARAMETERS	I	$w : V \rightarrow \mathbb{Q}$	node weights
VARIABLES	X	$\forall i \in V x_i \in \{0, 1\}$ (= 1 iff $i \in U$)	binary
OBJECTIVE	O	$\max \sum_{i \in V} x_i$	$f(x) = \sum_i x_i$
CONSTRAINTS	C	$\forall \{i, j\} \in E (x_i + x_j \leq 1)$	$L_{ij} = -\infty$ $g_{ij}(x) = x_i + x_j$ $U_{ij} = 1$

MP Solvers

<i>Class</i>	<i>Properties</i>		<i>Solver</i>
LP	f, g linear	no integr. constr. on X	CPLEX
MILP	f, g linear	some integr. constr. on X	CPLEX
NLP	f, g nonlinear	no integr. constr. on X	COUENNE
cNLP	f, g convex, $L = -\infty$	no integr. constr. on X	SNOPT
MINLP	f, g nonlinear	some integr. constr. on X	COUENNE
cMINLP	f, g convex, $L = -\infty$	no integr. constr. on X	BONMIN

- MP implementation of choice: AMPL (www.aml.com)
- MP solver of choice: CPLEX (www.ilog.com)
- Solving the previous example using AMPL+CPLEX, get optimal solution $U = \{1, 3, 5\}$

MP for platforming optimization

- Some constraints depend on *attribute type* and propagation function, i.e. depend on the specific scenario at hand

Example. *Weight* is propagated linearly

logical properties are propagated according to \wedge or \vee

- Other constraints are valid over all possible scenarios

Example. Each *elementary functionality* is allocated at most one *physical component*

- Use *abstract function symbols* $\psi, \delta, \varphi, \chi$ in scenario-specific constraints
- Obtain a “MP schema” depending on $\psi, \delta, \varphi, \chi$
- A MP schema cannot be solved; in order to solve it, we need to instantiate the abstract functions with concrete functions derived for the scenario at hand

The mathematical details

The first step is implemented by constraints (1) and (2): for each customer attribute h , the scalar (discrete) value yielded by the requirement function ψ_{vh}^i must belong to the interval (the set) of the relevant requirement.

$$\begin{aligned} \forall i \in P, v \in V, h \in F_s(v) \quad \Delta_{vh}^i &\leq \psi_{vh}^i(\mathbf{y}_v^i) \leq \bar{\lambda}_{vh}^i & (1) \\ \forall i \in P, v \in V, h \in F_d(v) \quad \psi_{vh}^i(\mathbf{y}_v^i) &\in \lambda_{vh}^i. & (2) \end{aligned}$$

The second step can be modeled by resorting to the composition functions δ_{vk} . The attribute values of a module depend on the attribute values of its selected submodules, see constraints (3), whereas the attribute values of an elementary module must correspond to those of the physical component used to implement it, see constraints (4).

$$\begin{aligned} \forall i \in P, v \in V \setminus L, k \in A \quad y_{vk}^i &= \delta_{vk}(\mathbf{y}_k^i \odot \mathbf{x}^i) & (3) \\ \forall i \in P, v \in L, k \in A(v) \quad y_{vk}^i &= \sum_{q \in I(v)} z_{vq}^i y_{qk}^i. & (4) \end{aligned}$$

Notice $\mathbf{y}_k^i \odot \mathbf{x}^i = (y_{ku_1}^i x_{u_1}^i, \dots, y_{ku_\alpha}^i x_{u_\alpha}^i)$ where $\alpha = |N(v)|$, and that each argument $y_{ku}^i x_u^i$ of δ_{vk} takes the value y_{ku}^i if submodule u is currently selected, and 0 otherwise.

The third step is implemented by constraints (5) and (6) which guarantee that the scalar and discrete attribute values of a chosen physical component belong to the relevant design bounds.

$$\begin{aligned} \forall i \in P, q \in Q, k \in A_s(v) \quad \vartheta_k^q &\leq y_{qk}^i \leq \bar{\vartheta}_k^q & (5) \\ \forall i \in P, q \in Q, k \in A_d(v) \quad y_{qk}^i &\in \vartheta_k^q. & (6) \end{aligned}$$

Finally, each selected elementary module must be implemented by exactly one physical component:

$$\forall i \in P, v \in L \quad x_v^i = \sum_{q \in I(v)} z_{vq}^i. \quad (7)$$

3.B.2 Interface implementation

The functional and physical interfaces mainly concern the compatibility between selected modules and components. Due the definition of interface functions (see §2.E), such compatibility can be simply modeled by logical implications: the functional module v (physical component q) cannot be selected, i.e., $x_v^i = 0$ ($z_{vq}^i = 0$), if the current module (component) selection and functional (physical) attribute setting are not compatible, i.e., $\varphi_v(\mathbf{y}^i \odot \mathbf{x}^i) = 0$ ($\chi_q(\mathbf{y}^i \odot \mathbf{x}^i) = 0$).

$$\begin{aligned} \forall i \in P, v \in V \quad x_v^i &\leq \varphi_v(\mathbf{y}^i \odot \mathbf{x}^i) & (8) \\ \forall i \in P, v \in L, q \in I(v) \quad z_{vq}^i &\leq \chi_q(\mathbf{y}^i \odot \mathbf{x}^i) & (9) \end{aligned}$$

where $\mathbf{y}^i \odot \mathbf{x}^i = (y_{v_1 k_1}^i x_{v_1}^i, y_{v_1 k_2}^i x_{v_1}^i, \dots, y_{v_2 k_1}^i x_{v_2}^i, \dots)$, with $v \notin \{v_1, v_2, \dots\}$.

3.B.3 Commonality evaluation



... are too horrendous to behold, I will spare you (trust me)

picture: www.phdcomics.com

The AMPL program



```

set P; set Q; set V; set W := V union Q; set Fs; set Fd; set I{L};
set Fall := Fs union Fd; set F{V} default {}; set As; set Ad;
set Aall := As union Ad; set A{W}; set E within {V,V} default {};
set N{v in V} := {u in V : (v,u) in E}; set L := {v in V : card(N[v]) = 0};
param module{V} symbolic; param component{Q} symbolic;
param lambdaL{P,v in V,F[v] inter Fs}; param lambdaU{P,v in V,F[v] inter Fs};
param lambda_card{v in V,F[v] inter Fd} integer, >= 0;
param lambda{i in P,v in V,h in F[v] inter Fd, 1..lambda_card[v,h]} binary;
param thetaL{u in W, A[u] inter As}; param thetaU{u in W, A[u] inter As};
param theta_card{u in W, k in Ad} integer, >= 0; param M > 0 default 1000;
param theta{u in W, k in Ad, 1..theta_card[u,k]} binary;
var y_C{i in P, u in W, k in As} <= thetaU[u,k], >= thetaL[u,k];
var y_I{i in P, u in W, k in Ad, t in 1..theta_card[u,k]} binary;
subject to constr_on_y_I {i in P, u in Q, k in Ad inter A[u]: k != 3} :
    sum {t in 1..theta_card[u,k]} y_I[i,u,k,t] = 1;
var w{P,P,Q,Aall} binary; var x{P,V} binary; var p{Q} binary;
var z{P,V,Q} binary; var r{P,Q} binary; include "linear.mod";
maximize commonality :
    sum{q in Q} p[q] + sum{i in P, j in P, q in Q, k in A[q] : i < j} w[i,j,q,k];
subject to leaf_attributes_C {i in P, v in L, k in A[v] inter As} :
    y_C[i,v,k] = sum{q in I[v]} zCy[i,v,k,q];
subject to leaf_attributes_I
    {i in P, v in L, k in A[v] inter Ad, t in 1..theta_card[v,k]} :
    y_I[i,v,k,t] = sum{q in I[v]} zIy[i,v,k,q,t];
subject to allocation {i in P, v in L} : x[i,v] = sum{q in I[v]} z[i,v,q];
subject to commonality10 {i in P, v in L, q in I[v]} : z[i,v,q] <= r[i,q];
subject to commonality11a {i in P, j in P, q in Q, k in A[q] inter As : i < j}:
    y_C[i,q,k] - y_C[j,q,k] <= M*(1-w[i,j,q,k]);
subject to commonality11a_I {i in P, j in P, q in Q, k in A[q] inter Ad:i < j}:
    (sum {t in 1..theta_card[q,k]} y_I[i,q,k,t]) -
    (sum {t in 1..theta_card[q,k]} y_I[j,q,k,t]) <= M*(1-w[i,j,q,k]);
subject to commonality11b {i in P, j in P, q in Q, k in A[q] inter As : i < j}:
    -y_C[i,q,k] + y_C[j,q,k] <= M*(1-w[i,j,q,k]);
subject to commonality11b_I {i in P, j in P, q in Q, k in A[q] inter Ad:i < j}:
    -(sum {t in 1..theta_card[q,k]} y_I[i,q,k,t]) +
    (sum {t in 1..theta_card[q,k]} y_I[j,q,k,t]) <= M*(1-w[i,j,q,k]);
subject to commonality12{i in P,j in P,q in Q,k in A[q]:i<j}:w[i,j,q,k]<=r[i,q];
subject to commonality13{i in P,j in P,q in Q,k in A[q]:i<j}:w[i,j,q,k]<=r[j,q];
subject to commonality14 {q in Q} : card(A[q]) * (sum{i in P} r[i,q] - 1) -
    sum{i in P diff {card(P)}, k in A[q]} w[i,i+1,q,k] <= M * (1-p[q]);
subject to commonality15 {q in Q} : sum{i in P} r[i,q] >= p[q];
include "laptop-psi.mod"; include "laptop-delta.mod";
include "laptop-chi.mod"; include "laptop-phi.mod";

```

← main file
whole MP: 6 files

```

ILOG CPLEX 11.010, options: e m b q use=1
CPLEX 11.0.1: mipdisplay=2
MIP Presolve eliminated 1027 rows and 337 columns.
MIP Presolve modified 724 coefficients.
Reduced MIP has 1879 rows, 1148 columns, and 4829 nonzeros.
Clique table members: 267.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: none, using 1 thread.
Root relaxation solution time = 0.02 sec.

```

	Nodes			Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Node	ItCnt	Gap
0	0	91.8333	32		91.8333	229	
0	0	91.0000	19		Cuts: 23	290	
* 0+	0			91.0000	91.0000	290	0.00%

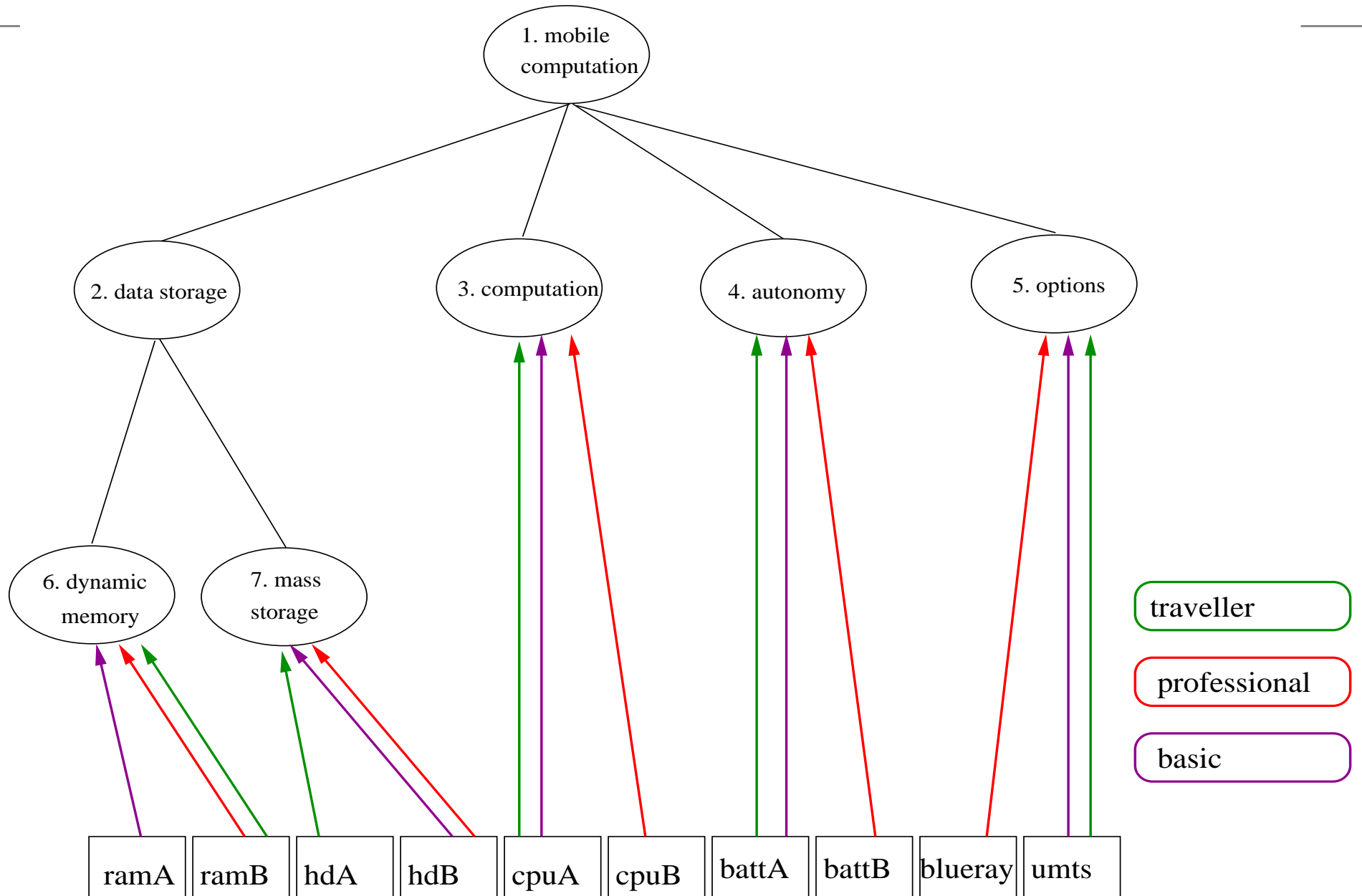
```

GUB cover cuts applied: 1
Gomory fractional cuts applied: 12
CPLEX 11.0.1: optimal integer solution; objective 91
290 MIP simplex iterations
0 branch-and-bound nodes
Tried aggregator 3 times
1 GUB-cover cut
12 Gomory cuts

```

Solved at the root node
of the BB tree

Solution



Conclusion

Contributions:

- Formally express the product family design problem
 - this allows the application of several different solution methods
- Make the **attribute** propagation problem independent by means of abstract functions
 - their implementation depends on the particular problem structure
- Provide a unified view of physical and functional architecture
 - as well as of module-bases and scale-based commonality
- Extend commonalization from physical components to functionalities