Optimal technological architecture evolutions of Information Systems[1]

Vassilis Giakoumakis[1] Daniel Krob[2] Fabio Roda[2] Leo Liberti[2]

[1] MIS, Université d Amiens, Amiens, France,

[2] LIX, École Polytechnique, 91128 Palaiseau, France

October 29, 2010

[1]Partially supported by École Polytechnique-Thales "Engineering of Complex Systems" Chair and ANR grant 07-JCJC-0151 "ARS".

Outline

Motivation

- Convergence (1): Systems Architecture, Optimization, Mathematical programming.
- Convergence (2): Enterprise Architecture and IT technical management perspectives
- A real problem (*"Kills"*): replacing some existing services with new services without impairing operations

Elements of information system architecture

Vassilis Giakoumakis, Daniel Krob, Fabio Roda, Leo Liberti

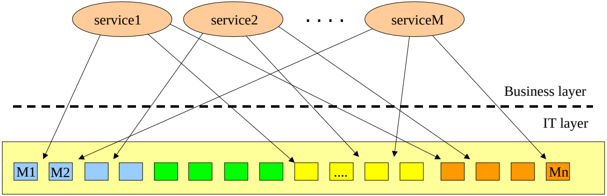


Figure: A simple two-layer information system architecture.

Elements of information system architecture

Any information system of an enterprise (consisting of a set D of departments) is classically described by two architectural layers:

- the **business layer**: the description of the business services offered by the information system;
- the **IT layer**: the description of the IT modules on which business services rely on.

In general, **the relationship between these two layers is not one-to-one**. A given business service can require any number of IT modules to be delivered and vice-versa a given IT module can be involved in the delivery of several business services.

Evolution of an information system architecture

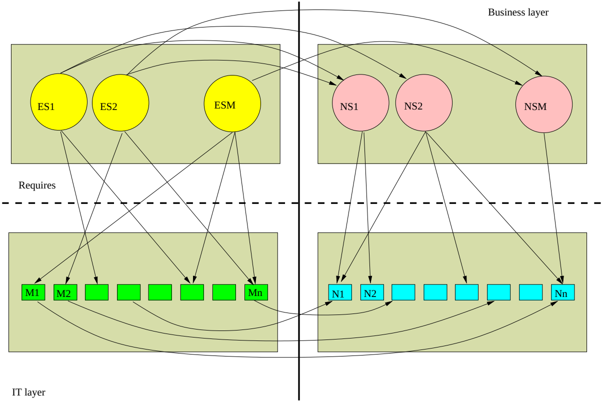


Figure: Evolution of an information system architecture.

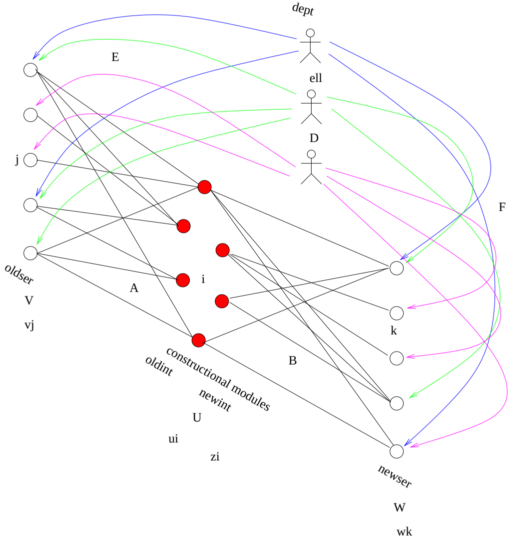
Evolution of an information system architecture

- From time to time **an information system may evolve** in its entirety.
- Strong impact at the IT layer level, where the existing IT modules $U^E = \{M_1, \ldots, M_n\}$ are **replaced** by new ones in a set $U^N = \{N_1, \ldots, N_{n'}\}$ (in the sequel, we assume $U = U^E \cup U^N$).
- This translates to a replacement of existing services (ES) by new services (NS) ensuring that the impact on the whole enterprise is kept low, to **avoid business discontinuity**.

Representation: Graph

Optimal technological architecture evolutions of Information Systems

Vassilis Giakoumakis, Daniel Krob, Fabio Roda, Leo Liberti



Graph

The enterprise consists of:

- a set D of departments;
- existing services V ;
- new services W .
- Each service relies on some IT module in U.
- The relations between services and modules and, respectively, departments and services, are denoted as follows:
  $A \subseteq V \times U$,
  $B \subseteq W \times U$,
  $E \subseteq D \times V$,
  $F \subseteq D \times W$.

Graph

- We use the **graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to model departments, existing services, new services, IT modules and their relations.
- The **vertices** are $\mathcal{V} = U \cup V \cup W \cup D$,
- the **edges** are $\mathcal{E} = A \cup B \cup E \cup F$.
- This graph is the **union of the four bipartite graphs** (U, V, A), (U, W, B), (D, V, E) and (D, W, F) encoding the respective relations. We remark that E and F collectively induce a relation between existing services and new services

Actors

Three main types of enterprise **actors** are naturally involved in the management of these technological evolutions which are described below.

1. **Business department managers**: they are responsible of creating business value through the new business services.
2. **IT project managers**: they are responsible for creating the new IT modules business services.
3. **Kill managers**: they are responsible for destroying the old IT modules in order to avoid to duplicate the information system — and therefore its operating costs — when achieving its evolution.

The information system architecture evolution management problem

- The department heads want to **maximize the value of the required new services**.
- The module managers want to **maximize the number of activated new modules according to an assigned schedule**
- The kill managers want to **maximize the number of deactivated old modules, within a certain kill budget**.

Optimization problem

Vassilis Giakoumakis, Daniel Krob, Fabio Roda, Leo Liberti

The rational planning of this evolution requires the solution of an optimization problem

Mathematical Programming

- **Mathematical Programming (MP)** is a formal language used for modelling and solving optimization problems
- Each problem is modelled by means of a list of index **sets**, a list of known **parameters** encoding the problem data (the *instance*), a list of **decision variables** an **objective function** to be minimized or maximized, and a set of **constraints**.
- A **solution** is an assignment of numerical values to the decision variables. A solution is **feasible** if it satisfies the constraints. A feasible solution is **optimal** if it optimizes the objective function.

Mathematical Programming based approach

Sets:

- $T = \{0, \ldots, t_{\max}\}$: set of time periods;
- U: set of IT modules;
- V: set of existing services;
- W: set of new services;
- $A \subseteq V \times U$: relations between existing services and IT modules;
- $B \subseteq W \times U$: relations between new services and IT modules;
- D: set of departments;
- $E \subseteq D \times V$: relations between departments and existing services;
- $F \subseteq D \times W$: relations between departments and new services.

Mathematical Programming based approach

Vassilis Giakoumakis, Daniel Krob, Fabio Roda, Leo Liberti

Parameters:

- $\forall i \in U \; a_i =$ cost of producing an IT module;
- $\forall i \in U \; b_i =$ cost of killing an IT module;
- $\forall k \in W \; c_k =$ revenue generated by a new service;
- $\forall t \in T \; H_t =$ production budget per time period;
- $\forall t \in T \; K_t =$ kill budget per time period;
- $\forall (i,k) \in B \; \beta_{ik} =$ monetary value given to new service k by IT module i.

Mathematical Programming based approach

Decision variables:

$$\forall i \in U, t \in T \; u_{it} = \begin{cases} 1 & \text{IT mod. (ES) is ON at t}= t \\ 0 & \text{otherwise;} \end{cases}$$

$$\forall i \in U, t \in T \; z_{it} = \begin{cases} 1 & \text{IT mod. (NS) is ON at t}= t \\ 0 & \text{otherwise;} \end{cases}$$

$$\forall j \in V, t \in T \; v_{jt} = \begin{cases} 1 & \text{old service } j \text{ is ON at t}= t \\ 0 & \text{otherwise;} \end{cases}$$

$$\forall k \in W, t \in T \; w_{kt} = \begin{cases} 1 & \text{new service } k \text{ is ON at t}= t \\ 0 & \text{otherwise.} \end{cases}$$

Mathematical Programming based approach

Objective function. Business value contributed to new services by IT modules.

$$\max_{u,v,w,y,z} \sum_{\substack{t\in T\\(i,k)\in B}} \beta_{ik} z_{it} w_{kt}.$$

Mathematical Programming based approach

Constraints.

- **Production budget (cost of producing new IT modules)**:

$$\forall t \in T \smallsetminus \{t_{\max}\} \quad \sum_{i \in U} a_i(z_{i,t+1} - z_{it}) \leq H_t,$$

  where the term $z_{i,t+1} - z_{it}$ is only ever 1 when a new service requires production of an IT module

- **Kill budget (cost of killing IT modules)**:

$$\forall t \in T \smallsetminus \{t_{\max}\} \quad \sum_{i \in U} b_i(u_{it} - u_{i,t+1}) \leq K_t,$$

  where the term $u_{it} - u_{i,t+1}$ is only ever 1 when an IT module is killed

Mathematical Programming based approach

- **Module activation**: once an IT module is activated, do not deactivate it.

$$\forall t \in T \smallsetminus \{t_{\max}\}, i \in U \quad z_{it} \leq z_{i,t+1}.$$

- **Module deactivation**: once an IT module is killed, cannot activate it again.

$$\forall t \in T \smallsetminus \{t_{\max}\}, i \in U \quad u_{it} \geq u_{i,t+1}.$$

- **Existing service**: if a ES is active, the necessary IT modules must also be active:

$$\forall t \in T, (i,j) \in A \quad u_{it} \geq v_{jt}.$$

- **New service**: if a NS is active the necessary IT modules must also be active:

$$\forall t \in T, (i,k) \in B \quad z_{it} \geq w_{kt}.$$

Mathematical Programming based approach

Vassilis Giakoumakis, Daniel Krob, Fabio Roda, Leo Liberti

- **Departments**: an existing service can be deactivated once all departments relying on it have already switched to new services:

$$\forall t \in T, j \in V \quad \sum_{k \in \mathcal{W}_j} (1 - w_{kt}) \leq |\mathcal{W}_j| v_{jt}.$$

- **Boundary conditions**: at $t = 0$ all IT modules needed by existing services are active, all IT modules needed by new services are inactive:

$$\begin{array}{lcl} \forall i \in U \ u_{i0} = 1 & \wedge & z_{i0} = 0; \\ \forall j \in V \ v_{j0} = 1 & \wedge & \forall k \in W \ w_{k0} = 0. \end{array}$$

- **Boundary conditions**: at $t = t_{\max}$ all IT modules needed by the existing services have been killed:

$$\forall i \in U \quad u_{it_{\max}} = 0.$$

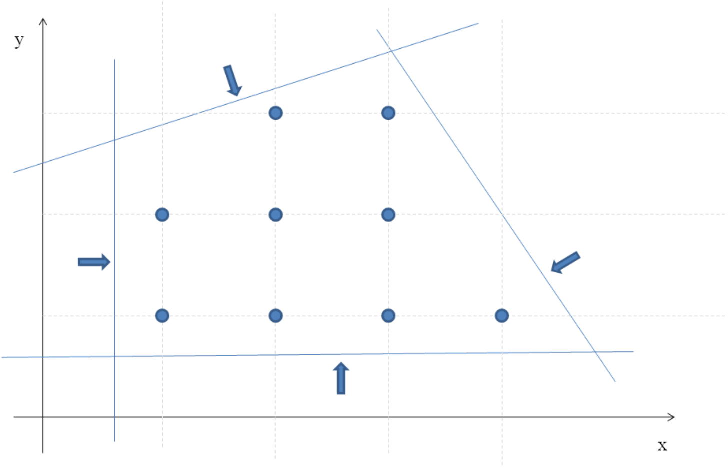
Branch and bound, ampl, cplex

The formulation above belongs to the MINLP class, as a product of decision variables appears in the objective function and all variables are binary; more precisely, it is a Binary Quadratic Program (BQP). This BQP can be solved directly using standard **Branch and bound** (BB) based solvers.
All our tests were performed using the **AMPL modelling environment** to implement the MP and the state-of-the-art off-the-shelf **CPLEX** to solve it.

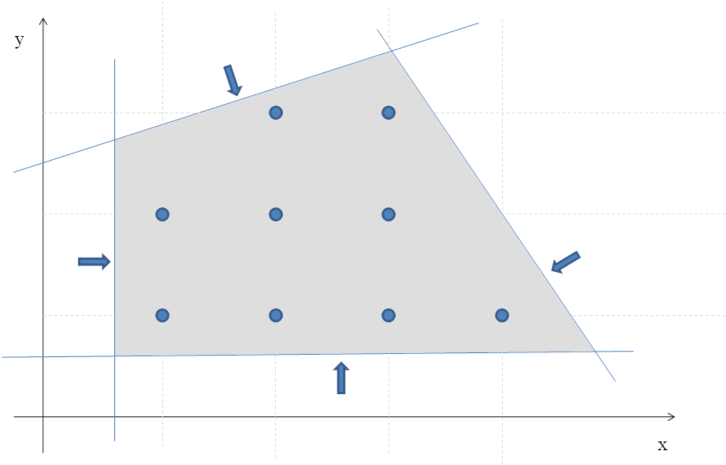
Cuts

- The BB method for for MPs with binary variables performs a binary tree-like recursive search. At every node, a **lower bound** to the optimal objective function value is computed by solving a continuous relaxation of the problem.
- The step of BB which **most deeply impacts its performance is the computation of the lower bound.**
- To improve the relaxation quality, one often adjoins **"redundant constraints"** to the problem whenever their redundancy follows from the integrality constraints. Thus, such constraints will not be redundant with respect to the relaxation.
- If an inequality is valid for an MP but not for its relaxation, it is called a **valid cut**.

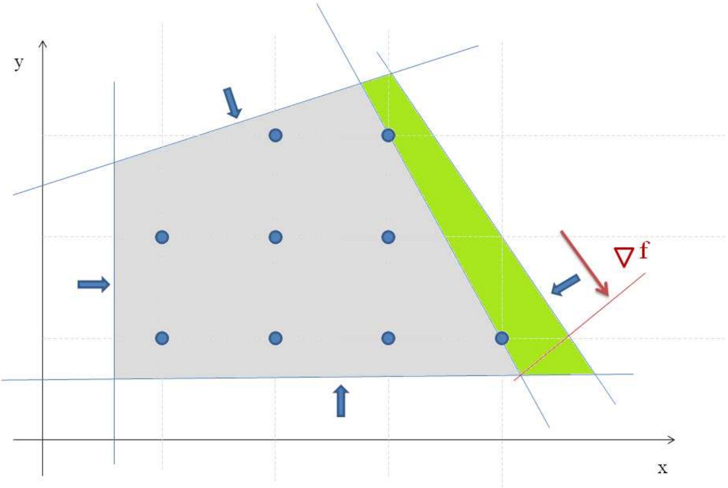
Cut: example



Cut: example



Cut: example



Cut: 1

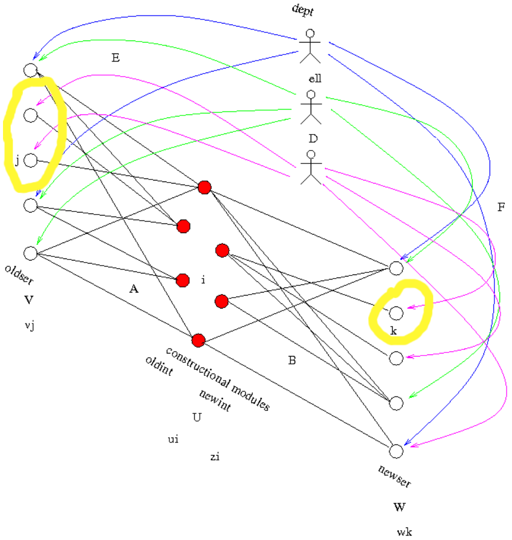
- Statement: *If a new service $k \in W$ is inactive, then all existing services linked to all departments relying on k must be active.*
- The statement corresponds to the inequality:

$$\forall t \in T, k \in W \quad \sum_{j \in \mathcal{V}_k} (1 - v_{jt}) \leq |\mathcal{V}_k| w_{kt}.$$

- (Remark) We formalize this statement by defining the sets:

$$\forall k \in W \quad \mathcal{V}_k = \{j \in V \mid \exists \ell \in D \; ((\ell, j) \in E \wedge (\ell, k) \in F)\}.$$

Cut: 1


dept
E
ell
D
j
F
oldser
V
vj
A
i
k
constructional modules
oldint
newint
U
B
ui
zi
newser
W
wk

Cut: 2

- Statement: *at any given time period no pair (ES, NS) related to a given department must be inactive (otherwise the department cannot be functional).*
- The statement corresponds to the inequality:

$$\forall t \in T, j \in V, k \in W \; \exists \ell \in D \; ((\ell, j) \in E \wedge (\ell, k) \in F)$$
$$v_{jt} + w_{kt} \geq 1$$

Computational Results

How we carried out our tests:

- We look at the CPU time and approximation guarantee behaviours in function of the instance size, and use these data **to assess the suitability of the method to real application.**
- Our results were obtained on a 64-bit 2.1 GHz Intel Core2 CPU with 4GB RAM running Linux.

Computational Results: instances

- we randomly generated **two sets** of 64 instances
- a set of **small instances** to be solved to guaranteed optimality
- a set of **large instances** where the BB algorithm is stopped either at BB termination or after 30 minutes.
- All instances have been randomly generated from a model that bears some similarity to the real instance data.
- The parameters of our model are in three categories: **cardinalities** (including all vertex sets), **graph density** (including all edge and arc sets), **monetary values** (including budgets).
- Each of the 64 instances in each set corresponds to a triplet (cardinality, edge creation probability, monetary value), each component of which ranges over a set of four elements.

Computational Results: small instances

- In order to observe how CPU time scales when solving to guaranteed optimality, we present 12 plots referring to the small set, grouped by row.
- we plot seconds of user CPU time
- for each **fixed cardinality**, in function of edge creation probability and monetary value (Fig. 3, first row);
- for each **fixed edge creation probability**, in function of cardinality and monetary value (Fig. 3, second row);
- for each **fixed monetary value**, in function of cardinality and edge creation probability (Fig. 3, third row).

Computational Results: small instances

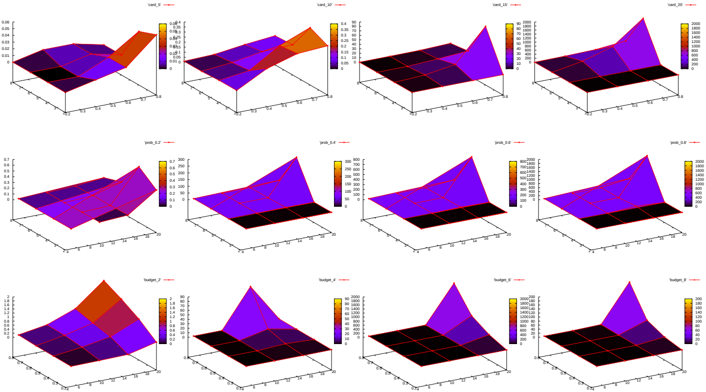


Figure: CPU time when solving small instances to guaranteed optimality.

Computational Results: small instances

- the proposed methodology **can solve a small instance to guaranteed optimality** within roughly half an hour;
- denser graphs and smaller budgets yield more difficult instances.
- Sudden drops in CPU time might correspond to infeasible instances, which are usually detected quite fast

Computational Results: large instances

- We plot the **optimality gap** — an approximation ratio — at termination rather than the CPU time, which is in this case limited to 30 minutes.
- The optimality gap, expressed in percentage, is defined as $\left(\frac{100|f^* - \bar{f}|}{|f^* + 10^{-10}|}\right)$ %, where f^* is the objective function value of the best feasible solution found within the time limit, and $\bar{f}$ is the tightest overall lower bound. A gap of 0% corresponds to the instance being solved to optimality.

Computational Results: large instances

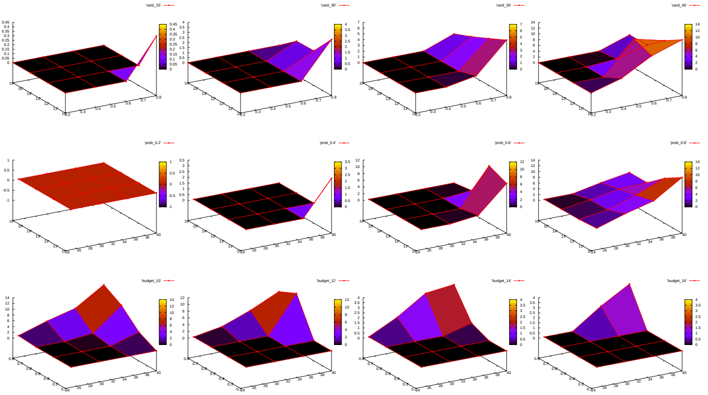


Figure: Optimality gap when solving large instances within 30 minutes of CPU time.

Computational Results: large instances

- The proposed methodology is able to solve large instances to a gap of 14% within half an hour of CPU time at worst, and to an average gap of 1.18% within an average CPU time of 513s (just over 8 minutes). target CPU time.

Conclusion

- We proposed a model for Architecture Evolutions of Information Systems
- We exhibited computational results showing that **an off-the-shelf** solver is capable of reaching a feasible solution with a satisfactory approximation guarantee within a realistic timeframe.
- Future work: Is the model realistic enough ?
- Thank you.