Technological architecture evolutions of Information Systems: trade-off and optimization*

Vassilis Giakoumakis† Daniel Krob‡ Leo Liberti‡ Fabio Roda‡

October 31, 2011

Abstract

In the normal lifespan of large enterprises, the strategic management of IT often evolves. Existing services must be replaced with new services without impairing operations. The problem of scheduling such replacement is of critical importance for the success of the operation. We analyze this problem from a quantitative point of view, underlining the trade-off nature of its solutions. We formalize this multi-objective optimization problem as a mathematical programming formulation. We discuss its theoretical properties and show that real-world instances can be solved by standard off-the-shelf tools.

1 Introduction

For any information system manager, a recurrent key challenge is to avoid creating more complexity within its existing information system through the numerous IT projects that are launched in order to respond to the needs of the business. Such an objective leads thus typically to the necessity of co-optimizing both creation *and* replacement/destruction — called usually *kills* in the IT language — of parts of the information system, and of prioritizing the IT responses to the business consequently.

This important question is well known in practice and quite often addressed in the IT literature, but basically only from an enterprise architecture or an IT technical management perspective [2, 5, 15]. Architectural and managerial techniques, however, are often only parts of the puzzle that one has to solve to handle these optimization problems. On the basis of budget, resource and time constraints given by the enterprise management, architecture provides the business and IT structure of these problems. This is however not sufficient to model them completely or solve them. Nevertheless, from a methodological point of view, real systems (especially information ones) can be described adopting different perspectives and different levels of details according to Systems Architecture methodologies that we can find in literature [3, 4, 12]. Some powerful techniques help to manage complex systems decoupling them into parts. *Abstraction* and *Concretization* are the processes that allow to shift from a specific level of details to a properly lower/higher one so that we can focus on the features which are really relevant and "hide" the others. *Decomposition* and *Integration* are the ones that let the analyst to identify and solve sub-problems. Alternating these activities we can provide a series of partial *visions* which cooperate to produce the whole picture, instead of a single global description.

According to these seminal concepts, in a previous work [10] we proposed a first abstract model (or *vision*) of the problem and moved a first step towards the integration of architectural business and IT

*This paper extends [10].

†MIS, Université d'Amiens, France. E-mail: vassilis.giakoumakis@u-picardie.fr.

‡LIX, École Polytechnique, 91128 Palaiseau, France. E-mail: dk,liberti,roda@lix.polytechnique.fr.

project management aspects. More precisely, we proposed an operational model and a Mathematical Programming (MP) formulation expressing a generic global prioritization problem occurring in the — limited, but practically relevant — context of a technological evolution of an information system, reducing to one all the different objectives of the different stakeholders.

In this paper we refine our analysis. In the interest of simplicity, the first model proposed understated the multi-objective nature of the problem. In particular the presence of many teams at work is worth a closer examination, because it can lead to puzzling situations if some specific conditions hold or resources are insufficient. If we introduce a bound on the duration of the whole transition, or, at least, on each step of it, and force the activation/deactivations to be done before a short deadline we generate a "competition" between departments. Thus, we employ multi-optimization techniques in order to model and numerically solve a wider part of this general problem. This is a second step towards a full formalization of the problem which promises to provide a valuable help for IT practitioners.

The rest of this paper is organized as follows: section 2 describes the problem and the key elements involved, section 3 proposes the Mathematical Programming formulation and introduces the necessary multi-optimization techniques, section 4 discusses the theoretical properties of the model and section 5 reports the results of the computational tests.

2 Operational model of an evolving information system

2.1 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 (forming a set V) offered by the information system;
- the *IT layer*: the description of the IT modules (forming a set U) on which business services rely on.

In general, the relationship $A \subseteq V \times U$ 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, as shown in Fig. 1.

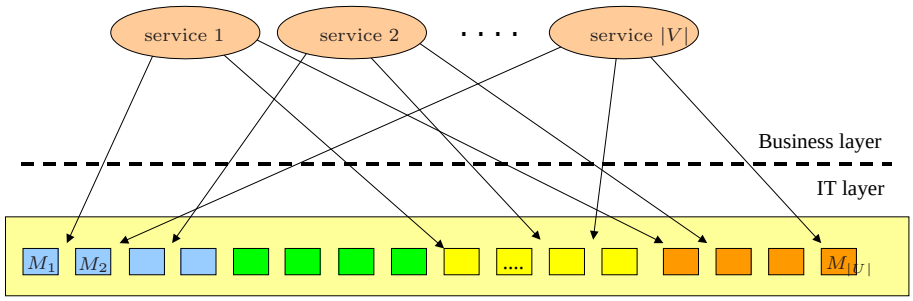

Figure 1: A simple two-layer information system architecture.

2.2 Evolution of an information system architecture

From time to time, an information system may evolve in its entirety due to the replacement of an existing software technology by a new one (e.g. passing from several independent/legacy software packages to an integrated one, migrating from an existing IT technology to a new one, and so on). These evolutions (or *transitions*) invariably have a 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 (sometimes denoted as ES) in V by new services (sometimes denoted as NS) in W ensuring that the impact on the whole enterprise is kept low in order to avoid business discontinuity. This also induces a relation $B \subseteq W \times U^N$ expressing reliance of new services on IT modules. Note also that in this context, at the business level, there exists a relation (in $V \times W$) between existing services and new services which expresses the fact that a given existing service shall be replaced by a subset of new business services. We note in passing that this relation also induces another relation in $U^E \times U^N$ expressing the business covering of an existing IT module to a subset of new IT modules (see Fig. 2).

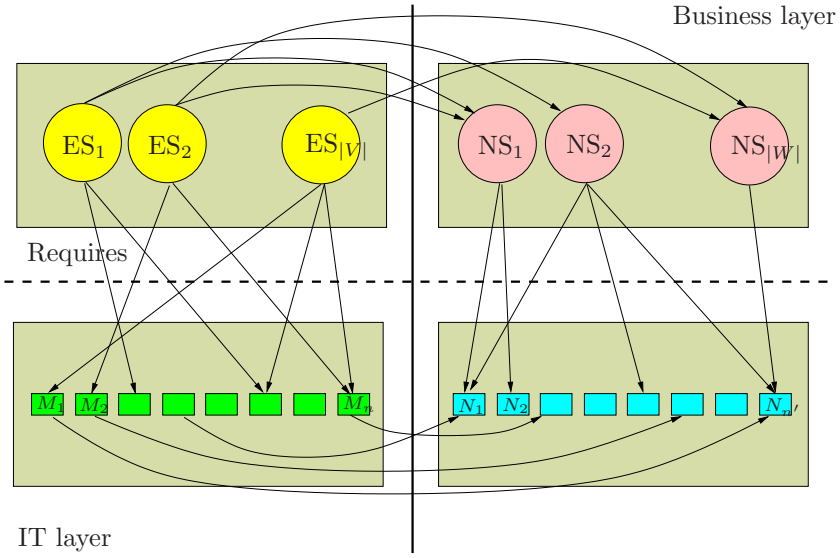

Figure 2: Evolution of an information system architecture.

2.3 Management of information system architecture evolutions

Mapping the above information system architecture on the organization of a company, it appears clear that 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 — within the perimeter of a business department in the set D — through the new business services. This value might be measured by the amount of money they are ready to invest in the creation of these services (business services are usually bought internally by their users within the enterprise).

2. *IT project managers*: they are responsible for creating the new IT modules, which is a pre-requisite to creating the associated business services. The IT project manager has a project schedule usually organized in work packages, each having specific starting times and global budget (see Fig. 2): in our case, this schedule is presented as a set of deployment precedence rules for new modules.

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. Kill managers have a budget for realizing such "kills", but they must ensure that any old IT module i is only killed after the new services replacing those old ones relying on i are put into service. The enterprise motivates the efficiency of kill managers by setting a monetary value on each deactivation: this provides a measure of the desirability of killing module i.

In this context, managing the technological evolution of an information system means being able of creating new IT modules within the time and budget constraints of the IT project manager in order to maximize both the IT modules business value brought by the new services and the associated kill value (i.e. the number of old services than can be killed).

2.4 The information system architecture evolution management problem

The architecture evolution of the IT system involves revenues, costs and schedules over a time horizon $t_{\max}$, as detailed below.

- *Time and budget constraints of the IT project manager.* Each new IT module $i \in U$ has a cost a_i and a production schedule.
- *IT module business value.* Each department $\ell \in D$ is willing to pay $q_{\ell k}$ monetary units for a new service $k \in W$ from a departmental production budget $H^\ell = \sum_{k:(\ell,k)\in F} q_{\ell k}$; the business value of the new service k is $c_k = \sum_{\ell:(\ell,k)\in F} q_{\ell k}$. We assume that this business value is transferred in a conservative way via the relation B to the IT modules. Thus, there is a business contribution β_{ik} over every $(i,k) \in B$ such that for each k we have $c_k = \sum_{(i,k)\in B} \beta_{ik}$; furthermore, the global business value of module i is $\sum_{k:(i,k)\in B} \beta_{ik}$. We also introduce a set $N \subseteq U$ of IT modules that are necessary to the new services.
- *Deployment schedule of new modules.* We are given a Directed Acyclic Graph (DAG) (U, S) where each couple $(i, h) \in S \subset U \times U$ encodes a deployment precedence between the new modules i, h (i.e. h cannot be deployed before i).
- *Kill value.* Discontinuing (or *killing*) a module $i \in U$ has a cost b_i due to the requirement, prior to the kill, of an analysis of the interactions between the module and the rest of the system architecture, in order to minimize the chances of the kill causing unexpected system behaviour. As mentioned above, it also has a monetary value (or desirability) ϕ_i .

The evolution involves several stakeholders. The department heads want to maximize the value of the required new services. The module managers want to produce the modules according to an assigned schedule whilst maximizing the business value for the new services to be activated. The kill managers want to maximize the monetary value of the deactivated modules within a certain kill budget. Thus, the rational planning of this evolution requires the solution of an optimization problem with several constraints and criteria, which we shall discuss in the next session.

3 Mathematical Programming based approach

Mathematical Programming (MP) is a formal language used for modelling and solving optimization problems [14, 19]. 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, which will contain appropriate values after the optimization process has taken place, an objective function to be minimized or maximized, and a set of constraints. The objective and constraints are expressed in function of the decision variables

and the parameters. The constraints might include integrality requirements on the decision variables. MPs are classified into Linear Programs (LP), Mixed-Integer Linear Programs (MILP), Nonlinear Programs (NLP), Mixed-Integer Nonlinear Programs (MINLP) according to the linearity of objective and constraints and to integrality requirements on the variables. MILPs and MINLPs are usually solved using a Branch-and-Bound (BB) method, explained at the beginning in Sect. 3.4. 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.

3.1 Multiobjective Programming

Multiobjective Programming (MOP) [9] is a modification of the MP language that allows for sets of objective functions to be supplied. Optimizing with respect to several objective functions at the same time makes the classical notion of optimum ill-defined. In such a setting, one looks at a domination relation between different feasible solutions. A feasible solution s dominates a second one s' (denoted by $s \prec s'$ if it is at least as good as the first one on all objectives, and strictly better on at least one objective. The set of all feasible non-dominated (or *efficient*) solutions is called the *efficient set*. The vector of objective function values corresponding to an efficient solution is called a *Pareto optimum*; the set of Pareto optima is called the *Pareto region* or *Pareto frontier* [1]. MOP is dealt with either by looking for the efficient set or by reformulating the MOP to a single-objective MP which identifies a single efficient solution. All efficient points are possible choices but the ultimate decision maker usually wants only one or a few suggestions. In this case the solution process for a MOP includes a first phase, which involves the computation of all efficient solutions, and a second one, which involves the selection of the most desirable efficient solution with respect to a set of preferences given by the decision maker. The role played by these preferences determines four approaches in literature: (*no-preference*, *"a posteriori"*, *"a priori"* and *interactive*), according to [16].

- *No-preference* methods do not use preferences and propose only one solution. In this case, the two phases are not clearly distinct; the final choice is based on a user-independent criterion.
- *"A priori"* methods make a limited use of preferences during the optimization process. Specifically, preferences are used to limit the extent of the efficient set found in the first phase.
- *"A posteriori"* methods search the whole efficient set and use preferences to delimit the final output given in the second phase. Typically, these methods are employed when we want to emphasize the presence of a trade-off rather than to provide a "small" solution set.
- *Interactive* methods mix different aspects of methods from previous categories.

In particular, we focus on two techniques that we use in the following. The L_p-metric method sketched below belongs the class of *no-preference* methods. It aims to identify a point in the objective space which has minimal distance from a reference point h^*. Typically, the chosen reference is the *ideal point (utopia)*, which corresponds to the maximum values of all objectives, optimized separately. In other words, the ideal point is the one we would chose if there were no trade-offs between tasks. For a vector $f = (f_1, \ldots, f_k)$ of objective functions, the ideal point is formally defined as the $h^* \in \mathbb{R}^k$ satisfying:

$$\forall i \in \{1, \ldots, k\} \quad h_i^* = \max\{f_i(s) \mid s \in X\}, \tag{1}$$

where depending on the decision variables s and X is the feasible region of the MOP. Figure 3 shows the sets of points that are equidistant from h^*, determined by different norms. The circle is given by the common Euclidean norm. The square and the rhombus are determined by the Maximum norm ($p = \infty$)

[1] According to [6] *"we have to remark that these notations are not unique in literature"* and that many authors do not distinguish between "efficient solution" and "Pareto optimum". Normally the meaning becomes clear by the context and they can be used synonymously (consider [6] for a discussion on this subject)

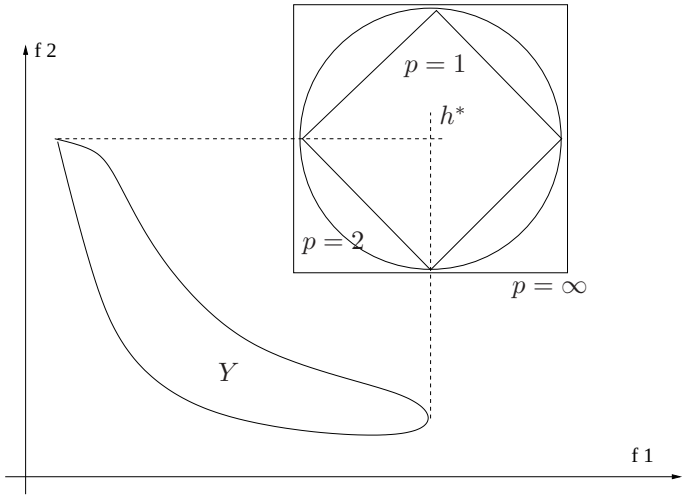

Figure 3: L_p-metric method

and the Manhattan norm ($p = 1$). Thus, if we use the utopia point as benchmark and the p-norm to define a metric, the general formulation of the method is:

$$\min \left(\sum_{i=1}^{k} |f_i(x) - h_i^*|^p \right)^{1/p} \tag{2}$$

We remark that if we consider $p = 1$, we obtain the single objective problem: $\min_s \sum_i |h_i^* - f_i(s)| = \min_s \sum_i (h_i^* - f_i(s)) = \|h^*\|_1 + \max_s \sum_i f_i(s)$, since $\|h^*\|_1$ is a constant. It therefore suffices to maximize $\sum_i f_i(s)$ with respect to s.

The weighted-sum method is (probably) the most common *"a posteriori"* approach to multi-objective optimization. Its principle is to assign to each individual objective function a weight $\alpha_i \geq 0$ normalized by $\sum_{i=1}^{k} \alpha_i = 1$ and then to replace the set of objective functions by the single compound objective $\min_s \sum_{i=1}^{k} \alpha_i f_i(s)$. By varying $\alpha = (\alpha_1, \alpha_2 .... \alpha_k)^T$ one can obtain a subset of the total set of efficient solutions (see Figure 4). If all of the weights are positive, the minimum of the scalar-valued formulation is an efficient solution. Conversely, there may be efficient solutions which cannot be found by minimizing any weight vector α (this might happen whenever the feasible set is nonconvex). In such cases, the weighted-sum method can be used to find an approximation of the efficient frontier. If necessary, other techniques can be applied in the second phase to inspect the objective space between solutions found by the weighted-sum method.

3.2 Introducing the MP formulation

As explained above, an enterprise in our context consists of a set D of departments currently relying on existing services in V and wishing to evolve to new services in W within a time horizon $t_{\max}$. Each service relies on some IT module in U. The set $N \subseteq U$ indexes those IT modules that are necessary. The relations between services and modules and, respectively, departments and services, are denoted as follows: $A \subseteq V \times U$, $B \subseteq W \times U$, $E \subseteq D \times V$ and $F \subseteq D \times W$. If an IT module $i \in U$ is required by a new service, then it must be produced (or activated) at a certain cost a_i. When an IT module $i \in U$ is no longer used by any service it must be killed at a certain cost b_i. Departments can discontinue using their existing services only when all new services providing the functionalities have been activated; when this happens, the service (and the corresponding IT modules) can be killed; a killed module i contributes ϕ_i monetary units to the goal of the kill manager. Departments have budgets dedicated to producing and killing IT modules, which must be sufficient to perform their evolution to the new services; for the

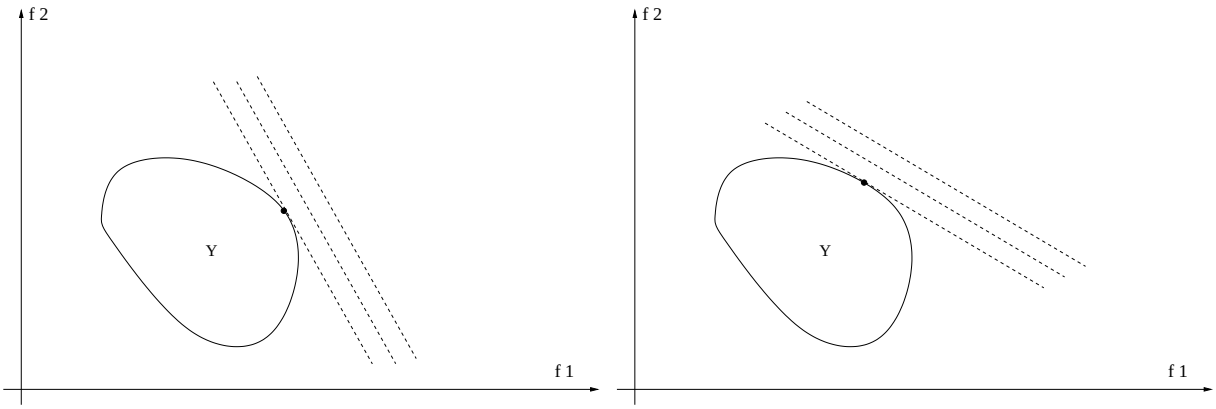

Figure 4: Weighted-sum method

purposes of this paper, we suppose that departmental budgets are interchangeable, i.e. all departments credit and debit their costs and revenues to two unique enterprise-level budgets: a production budget H_t and a kill budget K_t indexed by the time period t. A new service $k \in W$ has a value c_k, and an IT module $i \in U$ contributes β_{ik} to the value of the new service k that relies on it. We use the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ shown in Fig. 5 to model departments, existing services, new services, IT modules and their relations. The vertices are $\mathcal{V} = U \cup V \cup W \cup D$, and 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 with a "replacement" semantics (an existing service can be killed if the related new services are active).

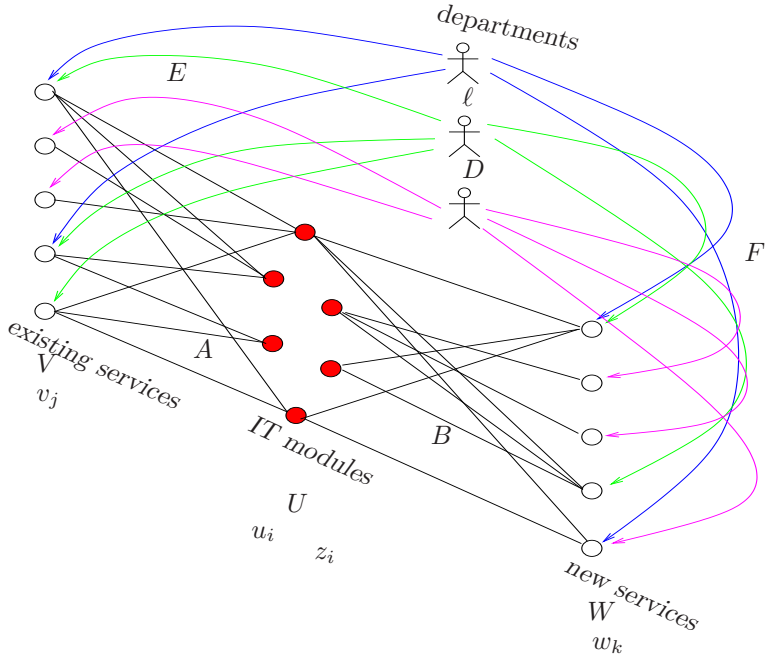

Figure 5: The bipartite graphs used to model the problem.

3.3 Sets, variables, objectives, constraints

We present here the MP formulation of the Architecture Evolution Problem (AEP). We recall that NS stands for new service and ES for existing service.

1. **Sets**:
   - $T = \{0, \ldots, t_{\max}\}$: set of time periods (Sect. 2.4, p. 4);
   - U: set of IT modules (Sect. 2.1, p. 2);
   - $N \subseteq U$: set of IT modules that are necessary for the NS (Sect. 2.4, p. 4);
   - V: set of existing services (Sect. 2.1, p. 2);
   - W: set of new services (Sect. 2.2, p. 3);
   - $A \subseteq V \times U$: relations between ES and IT modules (Sect. 2.1, p. 2);
   - $B \subseteq W \times U$: relations between NS and IT modules (Sect. 2.2, p. 3);
   - D: set of departments (Sect. 2.1, p. 2);
   - $E \subseteq D \times V$: relations between departments and ES (Sect. 3.2, p. 7);
   - $F \subseteq D \times W$: relations between departments and NS (Sect. 3.2, p. 7);
   - $S \subset N \times N$: deployment precedences between new modules (Sect. 2.4, p. 4).
2. **Parameters**:
   - $\forall i \in U\ a_i =$ cost of producing an IT module (Sect. 2.4, p. 2.4);
   - $\forall i \in U\ b_i =$ cost of killing an IT module (Sect. 2.4, p. 2.4);
   - $\forall i \in U\ \phi_i =$ desirability (monetary units) of killing an IT module (Sect 2.4, p. 2.4)
   - $\forall t \in T\ H_t =$ production budget per time period (Sect. 3.2, p. 3.2);
   - $\forall t \in T\ K_t =$ kill budget per time period (Sect. 3.2, p. 3.2);
   - $\forall (i,k) \in B\ \beta_{ik} =$ monetary value given to NS k by IT module i (Sect. 2.4, p. 2.4).
3. **Decision variables**:

$$\forall i \in U, t \in T\ u_{it} = \begin{cases} 1 & \text{if IT module } i \text{ is used for a ES at time } t \\ 0 & \text{otherwise;} \end{cases} \tag{3}$$

$$\forall i \in U, t \in T\ z_{it} = \begin{cases} 1 & \text{if IT module } i \text{ is used for a NS at time } t \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$

$$\forall j \in V, t \in T\ v_{jt} = \begin{cases} 1 & \text{if existing service } j \text{ is active at time } t \\ 0 & \text{otherwise;} \end{cases} \tag{5}$$

$$\forall k \in W, t \in T\ w_{kt} = \begin{cases} 1 & \text{if new service } k \text{ is active at time } t \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

4. **Objective functions**.
   - Business gain: value contributed to new services by IT modules. This is the objective of the module managers, and, because β is indexed on both modules and services, it agrees with the objective of the department heads.

$$\max_{u,v,w,z} \sum_{\substack{t \in T \\ (i,k) \in B}} \beta_{ik} z_{it} w_{kt}. \tag{7}$$

- Killing gain: objective of the kill managers.

$$\max_{u,v,w,z} \quad \sum_{\substack{t\in T\\ i\in U}} \phi_i(1-u_{it}). \tag{8}$$

5. **Constraints**.

- Production budget (cost of producing new IT modules; this is another objective of the module managers):

$$\forall t \in T \setminus \{t_{\max}\} \quad \sum_{i\in U} a_i(z_{i,t+1} - z_{it}) \leq H_t, \tag{9}$$

  where the term $z_{i,t+1} - z_{it}$ is only ever 1 when a new service requires production of an IT module — we remark that the next constraints prevent the term from ever taking value -1.

- Once an IT module is activated, do not deactivate it.

$$\forall t \in T \setminus \{t_{\max}\}, i \in U \quad z_{it} \leq z_{i,t+1}. \tag{10}$$

- Kill budget (cost of killing IT modules; this is part of the objective of the kill managers):

$$\forall t \in T \setminus \{t_{\max}\} \quad \sum_{i\in U} b_i(u_{it} - u_{i,t+1}) \leq K_t, \tag{11}$$

  where the term $u_{it} - u_{i,t+1}$ is only ever 1 when an IT module is killed — we remark that the next constraints prevent the term from ever taking value -1.

- Once an IT module is killed, cannot activate it again.

$$\forall t \in T \setminus \{t_{\max}\}, i \in U \quad u_{it} \geq u_{i,t+1}. \tag{12}$$

- If an existing service is active, the necessary IT modules must also be active:

$$\forall t \in T, (i,j) \in A \quad u_{it} \geq v_{jt}. \tag{13}$$

- If a new service is active, the necessary IT modules must also be active:

$$\forall t \in T, (i,k) \in B : i \in N \quad z_{it} \quad \geq \quad w_{kt}. \tag{14}$$

- An existing service can be deactivated once all departments relying on it have already switched to new services; for this purpose, we define sets $\mathcal{W}_j = \{k \in W \mid \exists \ell \in D\ ((\ell,j) \in E \wedge (\ell,k) \in F)\}$ for all $j \in V$:

$$\forall t \in T, j \in V \quad \sum_{k\in\mathcal{W}_j} (1 - w_{kt}) \leq |\mathcal{W}_j| v_{jt}. \tag{15}$$

- New modules must be deployed according to precedences: for a precedence $(i,h) \in S$, i must be deployed at least one timestep before h is; therefore, if $z_{it} = 0$ then $z_{hs} = 0$ for all $s \leq t$ (16), and if t is the first timestep where $z_{it} = 1$ then $z_{ht} = 0$ (17):

$$\forall s \leq t \in T, (i,h) \in S \quad z_{hs} - z_{it} \quad \leq \quad 0 \tag{16}$$

$$\forall s \leq t \in T \setminus \{0\}, (i,h) \in S \quad z_{hs} + z_{it} - z_{i,t-1} \quad \leq \quad 1. \tag{17}$$

- Boundary conditions. To be consistent with the objectives of the module and kill managers, we postulate that:

– at $t = 0$ all IT modules needed by existing services are active, all IT modules needed by new services are inactive:

$$\forall i \in U \setminus N \quad u_{i0} = 1; \tag{18}$$
$$\forall i \in N \quad u_{i0} = 0; \tag{19}$$
$$\forall i \in U \quad z_{i0} = 0; \tag{20}$$
$$\forall j \in V \; v_{j0} = 1 \quad \wedge \quad \forall k \in W \; w_{k0} = 0. \tag{21}$$

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

$$\forall i \in U \quad u_{it_{\max}} = 0. \tag{22}$$

These boundary conditions are a simple implementation of the objectives of module and kill managers. Similar objectives can also be pursued by adjoining further constraints to the MP, such as for example that the number of IT modules serving ES must not exceed a given amount.

The formulation above is a Binary Quadratic Program (BQP) with two objective functions. Single-objective BQPs exhibit products of binary decision variables [17]; they can either be solved directly using standard BB-based solvers [1, 11, 18] or reformulated exactly (see [13] for a formal definition of *reformulation*) to a MILP, by means of the ProdBin reformulation [7, 14] prior to solving is with standard MILP solvers.

3.1 Remark (Differences between single-objective and bi-objective formulations) *Consider the single objective formulation with objective function* (7) *only. This pursues the maximization of the business value by activating those modules that are necessary to the implementation of new services as soon as possible. The deactivation of old modules is considered as a cost, hence this formulation forces the solutions to respect a kill budget by means of constraint* (11)*. The time of deactivation might influence feasibility (through constraint* (15)*) but not the solution cost. In the biobjective formulation, on the other hand, objective function* (8) *induces an anticipation of the deactivation of the useless modules. Consider*

time	module: u_{it}	$u_{it} - u_{i,t+1}$	$1 - u_{it}$
1	1	0	0
2	1	0	0
3	1	0	0
4	1	1	0
5	0	0	1
...	...	...	...

time	module: u_{it}	$u_{it} - u_{i,t+1}$	$1 - u_{it}$
1	1	1	0
2	0	0	1
3	0	0	1
4	0	0	1
5	0	0	1
...	...	...	...

Table 1: Left: a partial solution of the single-objective formulation. Right: effect of second objective

the two partial solutions showed in Tables 1. We observe that $u_{it} - u_{i,t+1}$ *is positive only at step* 3*, while* $1 - u_{it}$ *is true from the deactivation till to the end. Thus the earlier the deactivation of a module occurs, the larger the value* $\sum_{i,t} \phi_i(1 - u_{it})$ *of objective* (8) *will be.*

3.4 Valid cuts from implied properties

The BB method 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. If all integral variables happen to take integer values at the optimum of the relaxation, the node is *fathomed* with a feasible optimum. If this optimum has better objective function value than the feasible optima found previously, it replaces the *incumbent*, i.e. the best current optimum. Otherwise, a variable x_j taking fractional value $\bar{x}_j$ is selected for branching. Two subnodes of the current node are created by imposing constraints $x_j \leq \lfloor \bar{x}_j \rfloor$ (left node) and $x_j \geq \lceil \bar{x}_j \rceil$ (right node) to the problem. If the

relaxed objective function value at a node is worse than the current incumbent, the node is also fathomed. 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. An inequality is *valid* for a MP if it is satisfied by all its feasible points. If an inequality is valid for an MP but not for its relaxation, it is called a *valid cut*.

We shall now discuss two valid inequalities for the evolution problem. The first one stems from the following statement: *If a new service $k \in W$ is inactive, then all existing services linked to all departments relying on k must be active.* We formalize this statement by defining the sets:

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

The statement corresponds to the inequality:

$$\forall t \in T, k \in W \quad \sum_{j \in \mathcal{V}_k} (1 - v_{jt}) \leq |\mathcal{V}_k| w_{kt}. \tag{23}$$

3.2 Lemma
Whenever (v, w) are part of a feasible solution of the evolution problem, (15) $\Leftrightarrow$ (23).

Proof. Firstly, we start proving that (15) $\Rightarrow$ (23). We proceed by contradiction: suppose (15) holds and (23) does not. Then there must be $t \in T, k \in W, j \in \mathcal{V}_k$ such that $w_{kt} = 0$ and $v_{jt} = 0$. By (15), $v_{jt} = 0$ implies $\forall h \in \mathcal{W}_j \ (w_{ht} = 1)$. By definition of $\mathcal{V}_k$ and $\mathcal{W}_j$, we have that $k \in \mathcal{W}_j$, and hence $w_{kt} = 1$ against the assumption. Secondly, we observe that the converse, (15) $\Leftarrow$ (23), also holds. The proof is symmetric: it suffices to swap j with k, $\mathcal{W}_j$ with $\mathcal{V}_k$, v with w, (15) with (23). □

We remark that (15) $\Rightarrow$ (23) let us assert that (23) is a valid inequality for the AEP. The second inequality is a simple relation between v and w.

3.3 Proposition
The inequalities

$$\forall t \in T, j \in V, k \in W \ \exists \ell \in D \ ((\ell, j) \in E \wedge (\ell, k) \in F) \quad v_{jt} + w_{kt} \geq 1 \tag{24}$$

are valid for the AEP.

Proof. Suppose (24) does not hold: hence there are $t \in T, j \in V, k \in W, \ell \in D$ with $(\ell, j) \in E$ and $(\ell, k) \in F$ such that $v_{jt} + w_{kt} = 0$. Since $v_{jt}, w_{kt} \geq 0$, this implies $v_{jt} = w_{kt} = 0$. It is easy to verify that if this is the case, (15) and (23) cannot both hold, contradicting (15) $\Leftrightarrow$ (23) (cf. Lemma (3.2)). □

Eq. (24) states that at any given time period no pair (ES, NS) related to a given department must be inactive (otherwise the department cannot be functional). We can add (23) and (24) to the MP formulation of the AEP, and hope they will improve the quality of the lower bound obtained via the LP relaxation. This is verified empirically in Sect. 5.3.

4 Formulation properties and trade-off

As mentioned previously, the formulation (3)-(22) models a biobjective problem. Notice, however, that the two objective functions (7), (8) involve different sets of variables. Thus, it is not immediately evident that this formulation might not be decomposed in two separate problems, one maximizing (7) and the other (8). Nor it is evident that the Pareto region might consist of more than one single optimum. In

other words, establishing whether this bi-objective formulation really corresponds to a trade-off type of problem is a relevant question.

In this section we present a mathematical analysis which shows that this is indeed the case. Specifically, we show that without (15), the problem can be decomposed in such a way that the only efficient solution is the utopia point. In this sense, (15) are the true source of the trade-off nature of (3)-(22). We first analyse a Lagrangian relaxation of (15), then give an example of a class of problem instances whose corresponding Pareto region fails to be a singleton set. For simplicity, we consider the original formulation only, without the valid cuts of Sect. 3.4.

4.1 Decomposability

Without (15), the formulation can be decomposed into two bi-objective subproblems involving only the u, v and respectively w, z variables. This suggests a Lagrangian relaxation [20] of (15) using Lagrangian coefficients $\lambda, \mu \geq 0$, which yields the two maximization objectives:

$$\begin{aligned}\zeta(\lambda,u,v,w,z) &= \sum_{\substack{(i,k)\in B\\ t\in T}} \beta_{ik} z_{it} w_{kt} + \sum_{\substack{t\in T\\ j\in V}} \lambda_{jt} \left(\sum_{k\in\mathcal{W}_j} (w_{kt}-1) + |\mathcal{W}_j| v_{jt} \right) = \\ &= \sum_{\substack{t\in T\\ (i,k)\in B}} \beta_{ik} z_{it} w_{kt} + \sum_{\substack{t\in T, j\in V\\ k\in\mathcal{W}_j}} \lambda_{jt} w_{kt} + \sum_{\substack{t\in T\\ j\in V}} |\mathcal{W}_j| \lambda_{jt} (v_{jt}-1),\end{aligned}$$

and

$$\begin{aligned}\eta(\mu,u,v,w,z) &= \sum_{\substack{t\in T\\ i\in U}} \phi_i u_{it} + \sum_{\substack{t\in T\\ j\in V}} \mu_{jt} \left(\sum_{k\in\mathcal{W}_j} (w_{kt}-1) + |\mathcal{W}_j| v_{jt} \right) = \\ &= \sum_{\substack{t\in T\\ i\in U}} \phi_i u_{it} + \sum_{\substack{t\in T, j\in V\\ k\in\mathcal{W}_j}} \mu_{jt} w_{kt} + \sum_{\substack{t\in T\\ j\in V}} |\mathcal{W}_j| \mu_{jt} (v_{jt}-1).\end{aligned}$$

Thus, we can decompose the problem of Sect. 3.2 into the two bi-objective Lagrangian subproblems P, Q:

$$\left.\begin{array}{rrcl} \max\limits_{u,v} & \sum\limits_{\substack{t\in T\\ j\in V}} |\mathcal{W}_j| \lambda_{jt} (v_{jt}-1) & & \\ \max\limits_{u,v} & \sum\limits_{\substack{t\in T\\ i\in U}} \phi_i u_{it} + \sum\limits_{\substack{t\in T\\ j\in V}} |\mathcal{W}_j| \mu_{jt} (v_{jt}-1) & & \\ \forall t \in T \smallsetminus \{t_{\max}\} & \sum\limits_{i\in U} b_i (u_{it} - u_{i,t+1}) & \leq & K_t \\ \forall t \in T \smallsetminus \{t_{\max}\}, i \in U & u_{it} & \geq & u_{i,t+1} \\ \forall t \in T, (i,j) \in A & u_{it} & \geq & v_{jt} \\ \forall i \in U & u_{i0} & = & 1 \\ \forall j \in V & v_{j0} & = & 1 \\ \forall i \in U & u_{it_{\max}} & = & 0, \end{array}\right\} \quad (25)$$

$$\left.\begin{array}{rll}
\max\limits_{w,z} & \sum\limits_{\substack{t\in T\\(i,k)\in B}} \beta_{ik} z_{it} w_{kt} + \sum\limits_{\substack{t\in T, j\in V\\ k\in\mathcal{W}_j}} \lambda_{jt} w_{kt} & \\
\max\limits_{w,z} & \sum\limits_{\substack{t\in T, j\in V\\ k\in\mathcal{W}_j}} \mu_{jt} w_{kt} & \\
\forall t \in T \smallsetminus \{t_{\max}\} & \sum\limits_{i\in U} a_i(z_{i,t+1} - z_{it}) \leq H_t & \\
\forall t \in T \smallsetminus \{t_{\max}\}, i \in U & z_{it} \leq z_{i,t+1} & \\
\forall t \in T, (i,k) \in B : i \in N & z_{it} \geq w_{kt} & \\
\forall s \leq t \in T, (i,h) \in S & z_{hs} - z_{it} \leq 0 & \\
\forall s \leq t \in T \smallsetminus \{0\}, (i,h) \in S & z_{hs} + z_{it} - z_{i,t-1} \leq 1 & \\
\forall i \in U & z_{i0} = 0 & \\
\forall j \in V & w_{j0} = 0. &
\end{array}\right\} \tag{26}$$

We remark that objective functions for (25) and (26) are specific instances of the following more general case:

$$\left.\begin{array}{ll}
\max\limits_{y\in Y} & f(y) + g(y) \\
\max\limits_{y\in Y} & g(y)
\end{array}\right\}. \tag{27}$$

for functions f, g and a set Y. We have the following results.

4.1 Proposition
The efficient set of (27) *is contained in that of:*

$$\left.\begin{array}{ll}
\max\limits_{y\in Y} & f(y) \\
\max\limits_{y\in Y} & g(y)
\end{array}\right\}. \tag{28}$$

Proof. Let y, y' be in the efficient set of (27). Then, either $f(y) + g(y) \leq f(y') + g(y')$ $(*)$ and $g(y) \geq g(y')$ $(\dagger)$, or $f(y) + g(y) \geq f(y') + g(y')$ $(**)$ and $g(y) \leq g(y')$ $(\ddagger)$. By rearrangement of $(*)$ we have $f(y) - f(y') \leq g(y') - g(y)$; by $(\dagger)$, $g(y') - g(y) \leq 0$. Therefore, $f(y) - f(y') \leq 0$. By rearrangement of $(**)$ we have $f(y) - f(y') \geq g(y') - g(y)$, which by $(\ddagger)$ is ≥ 0, hence $f(y) - f(y') \geq 0$. Thus y, y' are in the efficient set of (28). □

By Prop. 4.1 one could find the efficient sets of:

$$\left.\begin{array}{ll}
\max\limits_{u,v} & \sum\limits_{\substack{t\in T\\ j\in V}} |\mathcal{W}_j| \lambda_{jt}(v_{jt} - 1) \\
\max\limits_{u,v} & \sum\limits_{\substack{t\in T\\ i\in U}} \phi_i u_{it} \\
\multicolumn{2}{c}{\text{constraints of (25)}}
\end{array}\right\} \tag{29}$$

$$\left.\begin{array}{ll}
\max\limits_{w,z} & \sum\limits_{\substack{t\in T\\(i,k)\in B}} \beta_{ik} z_{it} w_{kt} \\
\max\limits_{w,z} & \sum\limits_{\substack{t\in T, j\in V\\ k\in\mathcal{W}_j}} \mu_{jt} w_{kt} \\
\multicolumn{2}{c}{\text{constraints of (26)}}
\end{array}\right\} \tag{30}$$

and then simply filter out all the dominated solutions with respect to the objectives of (25) and, respectively, (26).

Formulations (29)-(30) are difficult to solve because, in accordance with Lagrangian duality theory, one would also have to minimize with respect to λ, μ. In practice, one could employ a "pure decomposition" where $\lambda = \mu = 0$. This reduces (29)-(30) to the two following single-objective problems:

$$\left.\begin{array}{ll}
\max\limits_{u,v} & \sum\limits_{\substack{t\in T\\ i\in U}} \phi_i u_{it} \\
\multicolumn{2}{c}{\text{constraints of (25)}}
\end{array}\right\} \tag{31}$$

$$\left.\begin{array}{ll}
\max\limits_{w,z} & \sum\limits_{\substack{t\in T\\(i,k)\in B}} \beta_{ik} z_{it} w_{kt} \\
\multicolumn{2}{c}{\text{constraints of (26)}}
\end{array}\right\} \tag{32}$$

This proves the following result:

4.2 Theorem
Relaxing (15) *yields a MP with the single objective function* (7) + (8).

In other words, the MP cannot be decomposed unless constraints (15) are relaxed.

4.2 Trade-off

Often, in complex environments, architects and systems engineers have to regard many non-functional requirements such as safety and maintainability. These requirements do not change the main mission of a system which is to fulfil the functional requirements but influence its quality. Different architectures could attain the target, uncovering different additional features. The main functional requirement that the evolution of an information system must satisfy is the complete switching to new services without service discontinuity. This task is optimized when the transition can be done quickly, in order to profit from the income provided by the new services. The first objective of the model (business gain) expresses this feature. The second objective, killing gain, incorporates the need of keeping the system "clean", because it rewards the fact that old modules are removed. These useless modules pollute the system and may, in the worst case, introduce some elements of risk, if obsolete functionalities are not managed properly. Thus, this second objective pushes the research of transitions which leads to a configuration of the whole system which exhibits good non-functional attributes.

Consider the following class of instances: $U = \{1, 2\}$, $N = \{1, 2\}$, $V = \{1, 2\}$, $D = \{1, 2\}$, $W = \{3, 4\}$, $A = \{(1, 1), (2, 2)\}$, $B = \{(1, 3), (2, 4)\}$, $E = \{(1, 1), (2, 2)\}$, $F = \{(1, 3), (2, 4)\}$, summarized graphically in Fig. 6. Two IT modules have to be switched from old services to new services by activating/deactivating

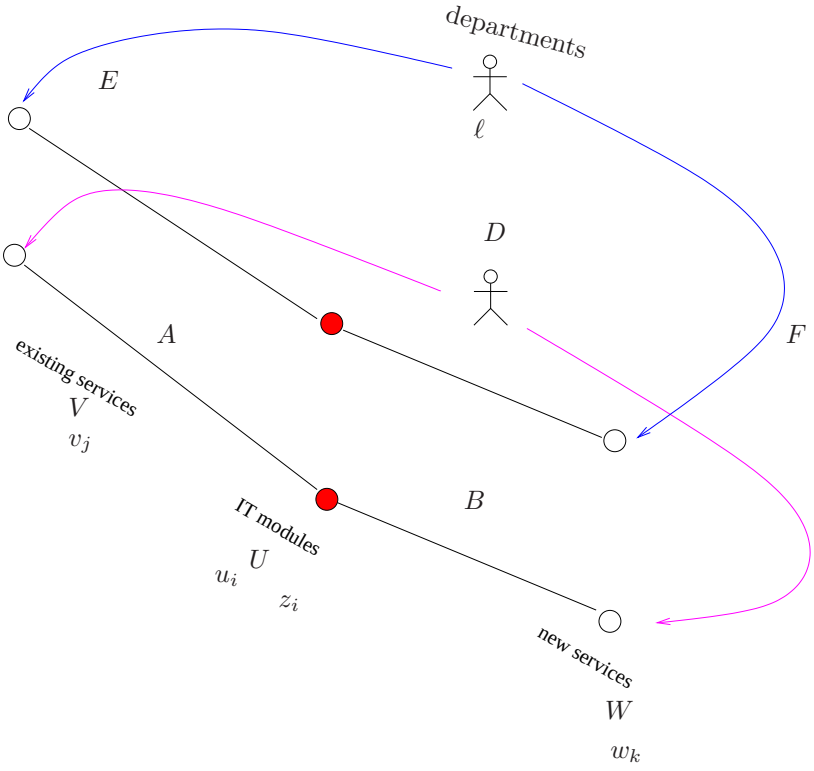

Figure 6: Basic instance

their interfaces to these services. The set T is limited to only two time periods. From this class, consider the specific instances given by the parameter values in Table 2.

We remark on some effects produced by the constraints of the model combined with the budget thresholds in Table 2. The production budget constraints (9) prevent both new modules from being switched on at the same time period, because this would cost 2 monetary units and the allowed bound

i	a_i	b_i	ϕ_i	β_{i1}	β_{i2}
1	0	1	2	0	0
2	0	1	1	0	0

t	H_t	K_t
0	0	0
1	1	1
2	1	1

Table 2: Time-independent (left) and time-dependent (right) parameter values

is 1. Similarly, the kill budget constraints (11) prevent both old modules from being switched off at the same time period. The utopia point therefore corresponds to the solution given in Table 3, where the most profitable modules are switched on/off first. Because of the values of ϕ and β, however, this solution

t	u_{1t}	u_{2t}	z_{1t}	z_{2t}	v_{1t}	v_{2t}	w_{1t}	w_{2t}
0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1
2	0	0	1	1	0	0	1	1

Table 3: Utopia point

is infeasible with respect to constraints 15.

These constraints require that when old modules are deactivated, and hence the corresponding services are stopped, the replacing (new) services, on which the departments rely, must already be in place. For example, if we switch the old module 1 off, the old service 1 based on it has to be halted. Thus, department 1, which loses its existing service 1, needs the new service 3, and consequently the module 1 has to be switched (and similarly for department 2 with old service 2 and new service 4 and the corresponding module 2). The partial solution $u_1 = 0 \wedge z_2 = 1$ is not possible because of constraints (15), and this make the utopical solution in Table 3 infeasible. The allowed partial solutions are $u_1 = 0 \wedge z_1 = 1$ and $u_2 = 0 \wedge z_2 = 1$. Thus, a feasible efficient transition is represented by table 4: at first the most profitable deactivation is realised and then the less profitable activation must be carried out to avoid service discontinuity for the first business department. A second feasible and efficient transition

t	u_{1t}	u_{2t}	z_{1t}	z_{2t}	v_{1t}	v_{2t}	w_{1t}	w_{2t}
0	1	1	0	0	1	1	0	0
1	0	1	1	0	0	1	1	0
2	0	0	1	1	0	0	1	1

Table 4: Feasible Solution 1

is represented by table 5: at first the most profitable activation is realised, which means that the most profitable deactivation must be delayed. These two solutions are feasible. Nevertheless no option is

t	u_{1t}	u_{2t}	z_{1t}	z_{2t}	v_{1t}	v_{2t}	w_{1t}	w_{2t}
0	1	1	0	0	1	1	0	0
1	1	0	0	1	1	0	0	1
2	0	0	1	1	0	0	1	1

Table 5: Feasible Solution 2

clearly better than the other one. Business gain and killing gain are different if you choose the first solution or the second one. If we simply sum them without preference, namely if we weight equally the objectives, we get the same gain with both solutions (9 monetary units), but the "composition" of the

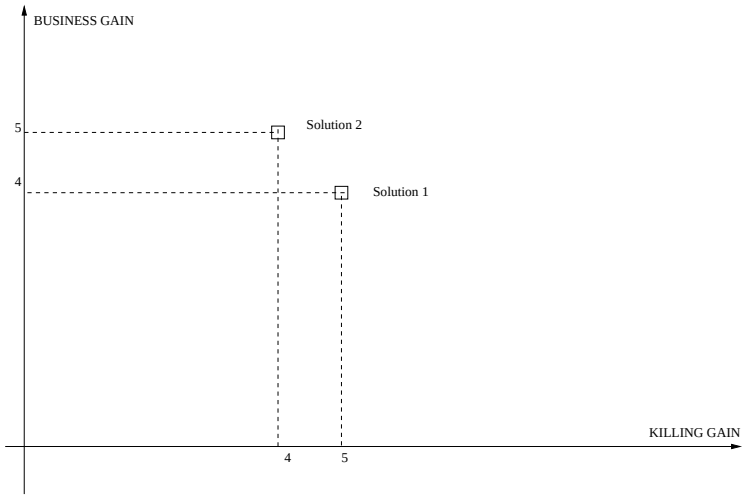

Figure 7: Basic trade-off

revenue varies. Figure 7 shows the Pareto region. We can observe that no solution dominates the other. If the decision maker favours the deactivation of the old modules, then the first solution is preferable. If the decision maker likes the activation of the new services better, the second solution becomes more desirable. This simple example dispels the doubt that this bi-objective problem might simply be a single objective problem in disguise, and highlights constraints (15) as the main source of the trade-off.

5 Computational results

In [10], which the present papers extends, we proposed a single objective model of the architecture evolutions problem and showed that it can be solved in a reasonable amount of time with regard to realistically sized instances. Here, we aim to establish if we can solve the MOP involving the two objectives (7) and (8) as well. We are more interested in evaluating the computational effort required rather than in exactly modelling the preferences of the decision makers. Hence, we adopt a no-preference approach, the L_p-metric method, with $p = 1$ and solve:

$$\min_{u,v,w,z} \left| \sum_{\substack{t\in T \\ (i,k)\in B}} \beta_{ik} z_{it} w_{kt} - h_1^* \right| + \left| \sum_{\substack{t\in T \\ i\in U}} \phi_i (1 - u_{it}) - h_2^* \right|, \tag{33}$$

where h_1^*, h_2^* are the optimum values of the single-objective maximizations of (7) and, respectively, (8), subject to all problem constraints.

We consider a set of small instances, to be solved to guaranteed optimality, and another set of larger instances where the BB algorithm is stopped either at BB termination or after 30 minutes of CPU time (whichever comes first). We use the AMPL modelling environment [8] and the off-the-shelf CPLEX 12.2 solver [11] running with its default configuration on a single 2.4 GHz Intel Xeon CPU with 8GB RAM. Ordinarily CPLEX's Quadratic Programming (QP) solver requires QPs with Positive Semi-Definite (PSD) quadratic forms only. Although in our case this may fail to hold, CPLEX can reformulate the problem exactly to the required form because all variables involved in the quadratic products are binary.

We consider the same set of instances both for the single objective form of the problem and the bi-objective form. All instances have been randomly generated from a model that bears some similarity to data coming from an actual service industry. We consider three parameter categories: cardinalities (vertex set), graph density (edge creation probability) and monetary values. 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.

Since our solution method of choice, which consists in solving (33), makes use of the single-objective optimum values h_1^* and h_2^*, we have to compute them. We remind that our previous paper [10] deals

Card.	Prob.	Bud.	feas.	cpu	Killing Gain
25	0.2	10	feasible	0.01	300
25	0.2	12	feasible	0.01	300
25	0.2	14	feasible	0.01	300
25	0.2	16	feasible	0.01	300
25	0.4	10	feasible	0.74	159
25	0.4	12	feasible	0.06	225
25	0.4	14	feasible	0.07	225
25	0.4	16	feasible	0.07	228
25	0.6	10	feasible	0.43	150
25	0.6	12	feasible	0.43	150
25	0.6	14	feasible	0.10	225
25	0.6	16	feasible	0.12	225
25	0.8	10	feasible	0.24	150
25	0.8	12	feasible	0.21	150
25	0.8	14	feasible	0.12	225
25	0.8	16	feasible	0.14	225
30	0.2	10	feasible	0.06	315
30	0.2	12	feasible	0.01	360
30	0.2	14	feasible	0.13	327
30	0.2	16	feasible	0.01	360
30	0.4	10	feasible	0.53	183
30	0.4	12	feasible	0.53	180
30	0.4	14	feasible	0.13	270
30	0.4	16	feasible	0.11	270
30	0.6	10	feasible	0.45	180
30	0.6	12	feasible	0.43	180
30	0.6	14	feasible	0.49	180
30	0.6	16	feasible	0.16	270
30	0.8	10	feasible	0.41	180
30	0.8	12	feasible	0.46	180
30	0.8	14	feasible	0.43	180
30	0.8	16	feasible	0.18	270
35	0.2	10	feasible	0.02	420
35	0.2	12	feasible	0.07	354
35	0.2	14	feasible	0.18	354
35	0.2	16	feasible	0.02	420
35	0.4	10	feasible	3.83	111
35	0.4	12	feasible	0.58	210
35	0.4	14	feasible	0.51	210
35	0.4	16	feasible	2.03	213
35	0.6	10	feasible	0.98	105
35	0.6	12	feasible	0.47	210
35	0.6	14	feasible	0.52	210
35	0.6	16	feasible	0.46	210
35	0.8	10	feasible	0.73	105
35	0.8	12	feasible	0.52	210
35	0.8	14	feasible	0.51	210
35	0.8	16	feasible	0.45	210
40	0.2	10	feasible	0.10	369
40	0.2	12	feasible	0.07	396
40	0.2	14	feasible	0.12	381
40	0.2	16	feasible	0.02	480
40	0.4	10	feasible	0.98	120
40	0.4	12	feasible	3.76	120
40	0.4	14	feasible	3.66	240
40	0.4	16	feasible	1.19	240
40	0.6	10	feasible	1.31	120
40	0.6	12	feasible	1.64	120
40	0.6	14	feasible	0.58	240
40	0.6	16	feasible	0.80	240
40	0.8	10	feasible	0.88	120
40	0.8	12	feasible	1.71	120
40	0.8	14	feasible	0.82	240
40	0.8	16	feasible	0.89	240

Table 6: Killing Gain - Single Objective

with the CPU time necessary for solving the mono-objective model which considers only the Business Gain and which provides the values of h_1^*. Thus, we focus now on CPU times necessary for solving the mono-objective model which considers only the Killing Gain and which gives the values of h_2^*. The results, for medium and big instances, are reported in Table 6. We remark that it can be achieved with small computational effort.

We suppose h_1^*, h_2^* as pre-calculated in the tests we present in the next section. Hence, our subsequent results do not consider the time needed to calculate the utopia point.

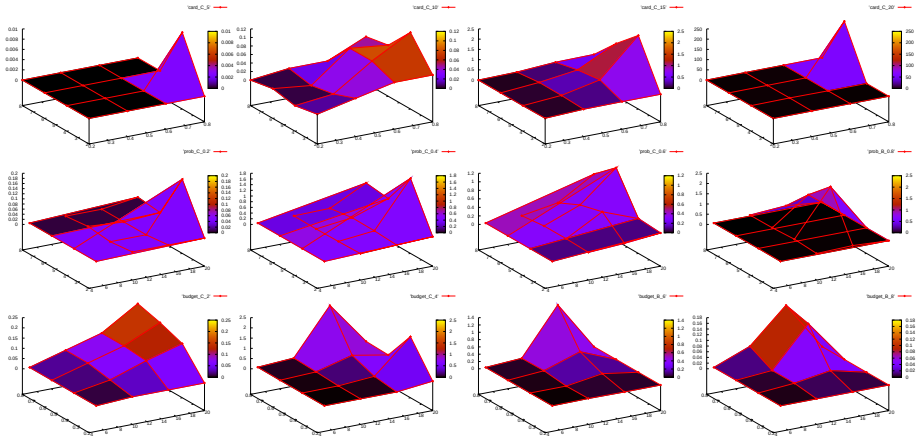

Figure 8: Bi-objective Model: CPU time when solving small instances.

5.1 CPU time

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. 8, first row); for each fixed edge creation probability, in function of cardinality and monetary value (Fig. 8, second row); for each fixed monetary value, in function of cardinality and edge creation probability (Fig. 8, third row). The largest "small instance" corresponds to the triplet $(20, 0.8, 8)$. The plots show that the proposed methodology can solve a small instance to guaranteed optimality within half an hour on standard computational equipment; it is also possible to notice that denser graphs and smaller budgets yield more difficult instances. Sudden drops in CPU time might correspond to infeasible instances, which, interestingly, are usually detected quite fast.

These results and the results reported in [10] show that we can solve both the single objective and the bi-objective formulations for realistic instances reasonably quickly. Table 7 shows the results of the comparison (with cardinality fixed at 20, i.e. the largest instances in the set of small instances) and the relative increase of the CPU time needed to solve the bi-objective formulation, with respect to the single-objective one (which considers only Business Gain). The effort is considerably higher but still manageable for practical purposes. Infeasibily is detected similarly in both models. Table 8 reports the results for medium and big instances. However in this case the timeout is set to 30 minutes and thus only executions shorter than 30 minutes are relevant to the comparison of CPU time. The other executions are relevant only for the evaluation of the optimality gap.

5.2 Optimality Gap

In Fig. 9 and Fig. 10 we plot the *optimality gap* at termination (which is limited to 30 minutes). The largest "large instance" corresponds to the triplet $(40, 0.8, 16)$. The optimality gap, expressed in percentage, is defined by CPLEX 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. The plots show that the proposed methodology is able to solve large instances to a gap of 12.8% within half an hour of CPU time at worst. It can solve and to an average gap of 1.13% both the single objective (which considers only Business Gain) and the bi-objective formulation (which considers both Business Gain and Killing Gain), within an average CPU time of 459s and 538s respectively (about 8 minutes). Table 8 reports the details of this comparison.

Edge Probability	Budget	feasible/infeasible	CPU time increment
0.4	2	infeasible	0.0
0.4	4	feasible	314.3
0.4	6	feasible	241.2
0.4	8	feasible	257.9
0.6	2	infeasible	12.5
0.6	4	infeasible	0.0
0.6	6	feasible	73.4
0.6	8	feasible	135.7
0.8	2	infeasible	-4.5
0.8	4	infeasible	4.0
0.8	6	feasible	186.7
0.8	8	feasible	137.0

Table 7: CPU time increment

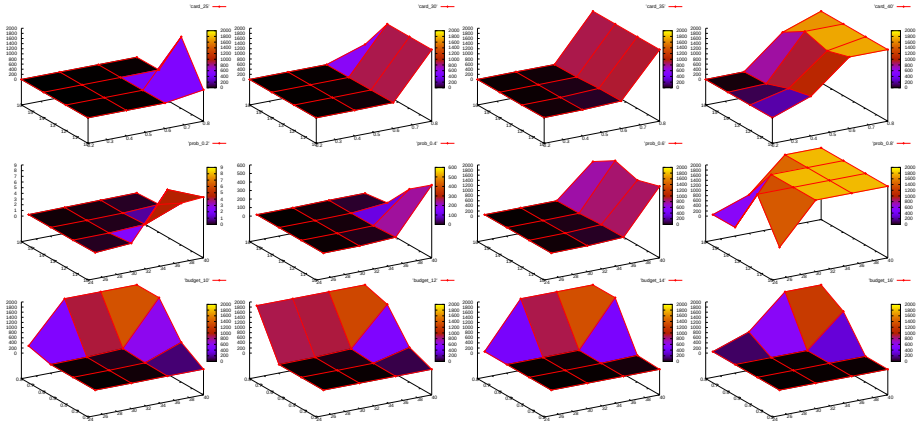

Figure 9: Single Objective Model: Optimality gap.

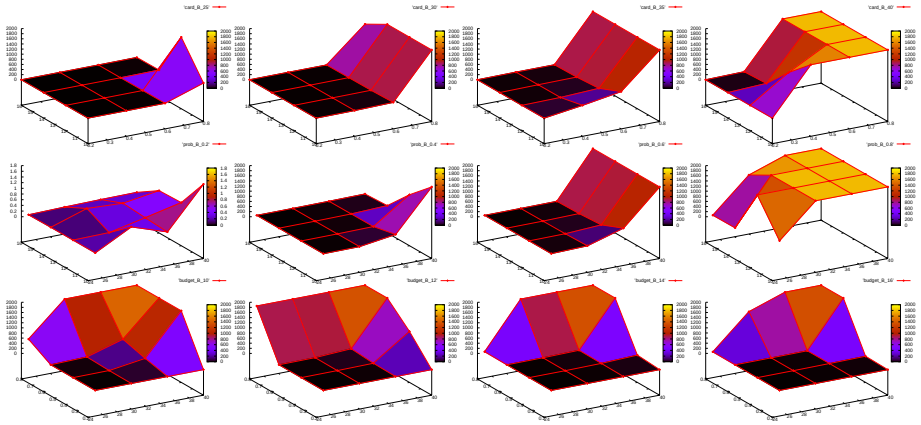

Figure 10: Bi-objective Model: Optimality gap.

5.3 Cuts effectiveness

As discussed in Section 3.4, valid cuts are redundant for the original formulation, but may improve the bound given by its continuous relaxation. In order to show that the cuts we have introduced are actually

Card.	Prob.	Bud.	feas. (1)	CPU (1)	obj. (1)	gap (1)	feas. (2)	CPU (2)	obj. (2)	gap (2)	CPU time increment
			Single Objective Model				Bi-objective Model				
25	0.20	10	yes	0.09	230	0.13	yes	0.03	529	0.37	-66.6
25	0.20	12	yes	0.03	261	0.58	yes	0.04	561	0.23	33.3
25	0.20	14	yes	0.02	190	0.65	yes	0.03	490	0.25	50.0
25	0.20	16	yes	0.02	205	0.06	yes	0.02	505	0.02	0.0
25	0.40	10	yes	0.12	344	1.45	yes	0.69	497	0.63	475.0
25	0.40	12	yes	0.17	404	0.21	yes	0.35	629	0.84	105.8
25	0.40	14	yes	0.04	411	1.62	yes	0.26	636	1.25	550.0
25	0.40	16	yes	0.1	424	3.41	yes	0.44	652	0.3	340.0
25	0.60	10	yes	1.33	424	0.42	yes	3.14	574	0.03	136.0
25	0.60	12	yes	0.46	499	0.37	yes	2.48	649	0.76	439.1
25	0.60	14	yes	0.22	578	1.13	yes	0.64	803	0.03	190.9
25	0.60	16	yes	0.32	562	0.79	yes	0.72	787	0.44	125.0
25	0.80	10	yes	196.76	540	0.01	yes	504.4	690	0.01	156.3
25	0.80	12	yes	1801.15	631	0.55	yes	1800.55	781	0.83	0.0
25	0.80	14	yes	5.85	734	0.02	yes	13.4	959	0.01	129.0
25	0.80	16	yes	2.15	745	0.08	yes	6.33	970	0.04	194.4
30	0.20	10	yes	0.51	304	0.05	yes	0.69	612	1.04	35.2
30	0.20	12	yes	0.05	344	0.59	yes	0.05	704	0.24	0.0
30	0.20	14	yes	0.04	303	1.05	yes	0.44	627	0.21	1000.0
30	0.20	16	yes	0.06	315	0.53	yes	0.05	674	0.47	-16.6
30	0.40	10	yes	2.54	477	0.09	yes	7.34	657	0.06	188.9
30	0.40	12	yes	0.95	532	0.14	yes	4.36	712	0.11	358.9
30	0.40	14	yes	0.27	609	0.49	yes	0.76	879	0.89	181.4
30	0.40	16	yes	0.26	560	0.16	yes	1.94	830	0.06	646.1
30	0.60	10	yes	13.27	635	0.01	yes	37.76	815	0.01	184.5
30	0.60	12	yes	10.06	634	0.02	yes	11.43	814	0.07	13.6
30	0.60	14	yes	3.11	734	0.06	yes	5.81	914	0.11	86.8
30	0.60	16	yes	0.31	824	0.19	yes	1.2	1094	0.34	287.1
30	0.80	10	yes	1800.96	772	3.84	yes	1800.46	952	3.55	0.0
30	0.80	12	yes	1800.96	789	1.31	yes	1800.45	969	1.44	0.0
30	0.80	14	yes	1800.99	837	1.73	yes	1800.46	1017	1.78	0.0
30	0.80	16	yes	359.42	1099	0.01	yes	1296.32	1369	0.01	260.6
35	0.20	10	yes	7.51	352	0.02	yes	0.28	764	0.02	-96.27
35	0.20	12	yes	0.31	377	0.04	yes	0.33	716	0.07	6.45
35	0.20	14	yes	0.09	425	0.11	yes	0.39	769	0.11	333.3
35	0.20	16	yes	0.09	437	0.03	yes	0.07	857	0.32	-22.2
35	0.40	10	yes	20.45	534	0.03	yes	212.1	641	0.02	937.1
35	0.40	12	yes	6.86	665	0.01	yes	20.34	875	0.01	196.5
35	0.40	14	yes	5.46	726	0.05	yes	15.19	936	0.04	178.2
35	0.40	16	yes	5.5	701	0.02	yes	10.72	914	0.07	94.9
35	0.60	10	yes	61.68	613	0.02	yes	459.1	718	0.01	644.3
35	0.60	12	yes	55.69	824	0.01	yes	87.78	1034	0.01	57.6
35	0.60	14	yes	12.79	816	0.02	yes	28.15	1026	0.01	120.0
35	0.60	16	yes	2.53	1012	0.13	yes	8.29	1222	0.03	227.6
35	0.80	10	yes	1800.83	579	7.74	yes	1800.45	684	7.32	0.0
35	0.80	12	yes	1800.85	978	4.97	yes	1800.4	1188	4.5	0.0
35	0.80	14	yes	1800.87	969	2.45	yes	1800.45	1179	2.33	0.0
35	0.80	16	yes	1800.72	1121	0.31	yes	1800.37	1331	0.56	0.0
40	0.20	10	yes	6.13	463	0.01	yes	1.72	812	0.04	-71.9
40	0.20	12	yes	2.75	453	0.01	yes	0.62	837	0.04	-77.4
40	0.20	14	yes	0.39	449	0.03	yes	0.59	822	0.01	51.2
40	0.20	16	yes	0.19	479	0.16	yes	0.08	948	0.1	-57.8
40	0.40	10	yes	511.06	588	0.01	yes	1800.31	708	1.38	252.2
40	0.40	12	yes	324.62	686	0.01	yes	994.54	806	0.01	206.3
40	0.40	14	yes	41.64	818	0.01	yes	69.13	1058	0.01	66.0
40	0.40	16	yes	7.48	906	0.03	yes	25.12	1146	0.02	235.8
40	0.60	10	yes	1800.88	638	2.36	yes	1800.33	758	6.72	0.0
40	0.60	12	yes	1273.08	753	0.01	yes	1800.27	873	2.9	41.4
40	0.60	14	yes	1800.95	1061	0.73	yes	1800.37	1301	0.66	0.0
40	0.60	16	yes	1222.44	1105	0.01	yes	1800.36	1345	0.37	47.2
40	0.80	10	yes	1800.72	720	12.84	yes	1800.33	840	11.49	0.0
40	0.80	12	yes	1800.75	807	9.18	yes	1800.46	927	8.3	0.0
40	0.80	14	yes	1800.72	1340	6.08	yes	1800.39	1580	5.48	0.0
40	0.80	16	yes	1800.74	1315	3.59	yes	1800.53	1555	3.41	0.0

Table 8: Comparison

useful, we compare the optimal solution value of the continuous relaxation with and without the cuts.

Table 9 reports the most interesting variations of the objective function we recorded during the tests. The value of the average variation for all the instances is 0.0106. Although this is may not

Card.	Prob.	Bud.	Objective function variation
25	0.4	10	0.012
30	0.2	10	0.016
35	0.2	12	0.011
40	0.2	10	0.018
40	0.2	14	0.019

Table 9: Cuts effectiveness

sound so impressive, one must bear in mind that these values refer to the root node relaxation only: improvements in deeper BB nodes might improve the bound considerably. Table 9 should only taken to be a counterexample dispelling the doubt that our cuts might be supposed useless.

5.4 Trade-off in realistic instances

We empirically observed a trend involving the edge density of the tripartite graph $(D \cup V \cup W, E \cup F)$ and the shape of the Pareto region. It is easy to see that, if $E \cup F = \varnothing$, then constraints (15) disappear, and hence the efficient set only consists of the utopia point. Pareto regions of different shapes and sizes can be obtained by employing instances with different edge sets $E \cup F$.

We consider medium-sized realistic instances (which correspond to triplets $(30, p, b(p))$, where vertex cardinality is fixed, the edge probability changes, and the budget is augmented with respect to the edge probability) and perform computational tests using the weighted-sum method. Varying the α coefficient vector, we obtain different points in the Pareto region. Fig. 11 shows three different Pareto regions,

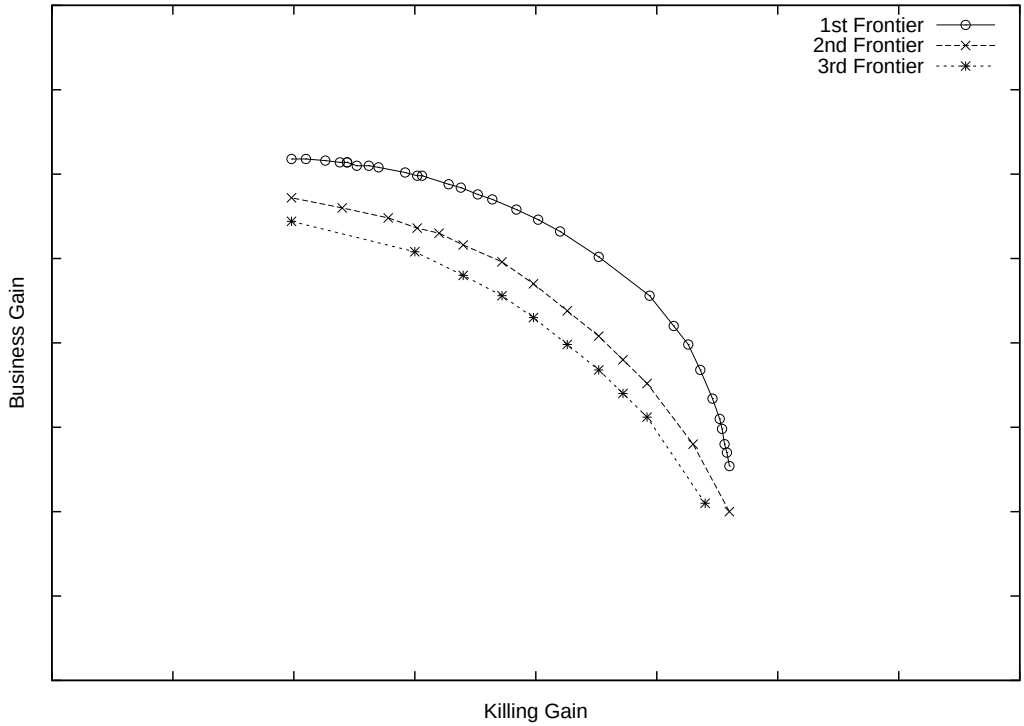

Figure 11: Pareto frontiers for realistic instances

corresponding to three different densities of the bipartite graphs linking departments with old and new services: densest graphs yield flatter Pareto regions, and vice versa.

6 Conclusion

The *information system architecture evolution management problem*, namely the problem of scheduling the replacement of existing services with new services without discontinuity, has considerable practical importance, yet was never previously formalized to the extent we discussed in this paper. The presence of many decision makers, as business department managers, IT project managers and kill managers, requires several objectives. These different stakeholders have different needs that the evolution of the system has to satisfy and this causes conflicts between the respective tasks, especially when the scheduling of the activities is tight. The decision makers typically aim to gain: (1) top business value produced by the new services, (2) the maximum number of new useful modules activated and (3) the maximum number of useless modules deactivated. In most situations, the objectives (1) and (2) are not really conflicting since

the activation of new services require new modules, thus Business and IT managers push the activities in the same direction. On the contrary, the objective (3) is potentially controversial, when there is a lack of time and resources. The activation of new modules and the deactivation of old ones requires work. If the amount of workforce is limited, as is usually the case, we then have to decide what has to be done first and, eventually, what is not necessary and can be planned for a later period. Business and project managers on one side and kill managers on the other have to compete for the existing resources and employ them for diverging aims. The former can fully attain their tasks on time only forcing the latter to delay theirs and vice versa. In our work, we define a Mathematical Programming formulation that models the problem correctly, provide a theoretical analysis thereof, showing exactly where the source of trade-off lies, and verify empirically that it can be used as a practical tool to solve realistically-structured instances.

Acknowledgments

This research was partially supported by the École Polytechnique Chairs "Engineering of Complex Systems" (Thalès) and "Optimization and Sustainable Development" (Microsoft-CNRS). We are grateful to two anonymous referees for their insightful comments.

References

[1] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.

[2] P. Bernus, K. Mertins, and G. Schmidt. *Handbook on Architectures of Information Systems*. Springer, Berlin, 2006.

[3] S. Bliudze and D. Krob. Towards a functional formalism for modelling complex industrial systems. In P. Bourgine, F. Kepes, and M. Schoenauer, editors, *European Conference on Complex Systems*, 2005.

[4] S. Bliudze and D. Krob. Towards a functional formalism for modelling complex industrial systems. *ComPlexUs, special Issue : Complex Systems - European Conference 2005*, 2(3-4):163–176, 2006.

[5] Y. Caseau. *Performance du système d'information – Analyse de la valeur, organisation et management (in French)*. Dunod, Paris, 2007.

[6] M. Ehrgott. *Multicriteria Optimization*. Springer, New York, 2005.

[7] R. Fortet. Applications de l'algèbre de Boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, 4:17–26, 1960.

[8] R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.

[9] A. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

[10] V. Giakoumakis, D. Krob, L. Liberti, and F. Roda. Optimal technological architecture evolutions of information systems. In M. Aiguier et al., editor, *Complex Systems Design and Management*, pages 137–148, Berlin, 2010. Springer.

[11] IBM. *ILOG CPLEX 12.2 User's Manual*. IBM, 2010.

[12] Daniel Krob. Modelling of complex software systems: A reasoned overview. In *FORTE*, pages 1–22, 2006.

[13] L. Liberti. Reformulations in mathematical programming: Definitions and systematics. *RAIRO-RO*, 43(1):55–86, 2009.

[14] L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations of Computational Intelligence Vol. 3*, number 203 in Studies in Computational Intelligence, pages 153–234. Springer, Berlin, 2009.

[15] J.N. Luftman. *Competing in the Information Age*. Oxford University Press, Oxford, 2003.

[16] K. Miettinen. *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, Boston, 1999.

[17] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming*, 45:139–172, 1989.

[18] N.V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2005.

[19] H.P. Williams. *Model Building in Mathematical Programming*. Wiley, Chichester, 4th edition, 1999.

[20] L.A. Wolsey. *Integer Programming*. Wiley, New York, 1998.