Multi-criteria improvement of complex systems

Jacky Montmain*, Christophe Labreuche², Abdelhak Imoussaten¹, and François Trousset¹

¹LGI2P, EMA, Parc Scientifique Georges Besse, 30035 Nîmes, France
{abdelhak.imoussaten,francois.trousset,jacky.montmain}@mines-ales.f,
²Thales Research & Technology, 1 avenue Augustin Fresnel, 91767 Palaiseau Cedex, France
{christophe.labreuche}@thalesgroup.com

Abstract
Designing the way a complex system should evolve to better match the customers’ requirements provides an interesting class of applications for multicriteria techniques. The required models to support the improvement design of a complex system must include both preference models and system behavioral models. A MAUT model captures the decisions related to customers’ preferences whereas a fuzzy representation is proposed to model the relationships between systems parameters and performances to capture operational constraints. The way in which these models are jointly used within our entire design procedure is intended to prove that both models must be used in tandem in order to address the managerial and implementation issues involved in an improvement project. The iterative improvement process is supported by a mathematical model, in addition to a software tool that allows testing our approach on an industrial case study. The search of adequate parameters part of the improvement design is supported by a branch and bound algorithm to compute the most relevant actions to be performed. Experiments emphasize the efficiency of the algorithm.

Keywords: Multiple criteria analysis, Multicriteria improvement, Constraint Satisfaction Problems, Industrial performance, Management strategies, Qualitative model, Approximate reasoning, Choquet fuzzy integral

1 Introduction
To satisfy a fluctuating demand and achieve a high level of quality and service, industries must develop and integrate new features in order to become

*Corresponding author. Tel.: +33 4 66 38 70 33; fax: +33 4 66 38 70 74.
or remain market leaders [30]. To deal with the complexity of current industrial contexts, new management strategies intended to bring about continuous improvement must take into account two imperatives: complex systems need to be adapted to an evolving context; and improvement assessment proves to be a thorny issue due to its dependence on multiple decisional aspects [2, 13]. When designing improvement measures for complex systems, multiple decisions need to be considered [1]. Examples include military information architecture [34] or industrial device performance improvement [3]. Such settings increase the complexity in multidisciplinary design regarding the fulfillment of functional, technical, environmental, economic and security requirements. In this context, industries focus more intently on optimization and evaluation activities during the design process in order to improve and adapt complex systems.

When company designers/operators choose a new architecture to improve their system, they must first check whether their solutions do not violate any constraints and whether they satisfy not only customer needs and technical specifications, but also the company’s strategic goals and interests. Furthermore, these issues are obviously not devoid of budgetary constraints. One may think of transforming this setting in one global optimization problem where the variables are the values of system parameters. This is very complex as the mapping from the parameters values to the fulfillment of strategic goals is not known explicitly. Moreover, industries are more inclined to accept continuous improvement on their system rather than a more thorough perturbation of system parameters. We propose thus in this work to proceed by successive improvements from an existing situation. We propose a set of possible actions on the system, where each action affects the value on one or several system parameters.

In order to bypass the difficulty of the unknown mapping from the parameters values to the fulfillment of strategic goals, the idea is to use a behavioral model. Such a model already exists in different domains: engineering (e.g. [21, 15, 14]), industry design [6], qualitative Bayesian networks [32]. As we are ultimately interested in improvements, we propose a behavioral model that relates the actions to improvements or degradations on the goals. This influence model is provided by experts and expresses their experience. Even if in theory, two experts may come up with two different models, we do not expect large discrepancies at this stage. Apart from the influence model, two other inputs are also necessary to relate the actions to the overall impact on the satisfaction of the system. The first one synthesizes the results of the influence model. Assume that two actions are performed; the first one improves a goal and the second one hurts the same goal. Then what is the overall impact on the goal? This depends on the attitude of the decision maker towards risk. The first input describes this attitude. The second input weighs up goals and produces an overall satisfaction of the system. It is based on a multi-criteria model to aggregate the goals. These two inputs
are subjective and represent the preferences of the decision maker.

In order to identify the set of actions that allows the decision maker to improve the overall satisfaction of the system in the most efficient way, we propose to handle separately the multi-criteria model and the influence model (and its synthesis). First of all (at the strategic level), we start by identifying the goals on which it will be more rewarding to improve the system. Then (at the operational level), we aim at finding, thanks to a branch-and-bound algorithm the set of actions that will improve these goals as much as possible, at the minimum cost. If there does not exist such a set of actions, the first step generates other goals to be improved and the optimization algorithm is launched once more.

This paper is organized as follows. Section 2 sets forth a formal model for the problem of interest herein. It begins by modeling the search for outputs to be improved as a multi-criteria optimization problem, before integrating this proposal into an iterative system improvement procedure. A general algorithm, based on two functions (FindCoalitions and FindActions) is proposed. Afterwards, Section 3 describes function FindCoalitions: it identify the coalition of goals/criteria to be improved first. Section 4 describes the influence model and the subjective model to synthesize its results. A branch-and-bound algorithm (function FindActions) implements in Section 5 as an efficient solution step. Section 7 proposes a case study inspired from the adaptive management of a manufacturing plant. Finally, section 8 discusses some of the works related to improving the competing architectures available in a context of multidimensional assessment.

2 Description of the optimization algorithm

2.1 List of concepts

For starters, a complex system is characterized by input parameters \( \gamma_1, \ldots, \gamma_p \), e.g. the accurate definition of all entities in the military force and their ties, or the industrial device control parameters. The set of all possible parameter vector values \((\gamma_1, \ldots, \gamma_p)\) is denoted by: \( \Gamma = \Gamma_1 \times \cdots \times \Gamma_p \). A system is thus defined by an element \( \gamma \in \Gamma \). Not all elements of \( \Gamma \) lead to admissible systems for the customer since some customer requirements must generally be met. The set of elements \( \Gamma_{Adm} \subseteq \Gamma \), for which the associated system satisfies these requirements, yields the feasible input parameter values.

The company that designs or operates a complex system must therefore first satisfy the functional requirements of its customers. All elements of \( \Gamma_{Adm} \) satisfying customer requirements are not however indistinguishable from the standpoint of the designer and customer. The company has its own set of goals, priorities and strategic reasons that may favor choosing one system over another; the company then needs to construct a preference model based on criteria that capture its own goals but also the customer satisfac-
tion. Such criteria behave in a way that refines the customer’s requirements within the improvement design process. The selection of a “best” solution for the company from among the set of satisfactory solutions for the customer then becomes a matter of strategic decision-making.

The set of attributes relative to the decision-making criteria are denoted: \( X_1, \ldots, X_n \). These are system’s observable outputs. We set \( N = \{1, \ldots, n\} \). Each vector of attribute values \((x_1, \ldots, x_n)\) represents a different alternative; the set of alternatives is then: \( X = X_1 \times \ldots \times X_n \). For a military architecture, these attributes serve to quantify the way the operational mission proceeds; they are obtained by large-scale simulations on architecture labs [34]. For industrial processes, these attributes might be: the work-in-progress level, flow synchronization, suppliers’ service rate, etc. [2].

Let \( T: \Gamma_{Adm} \rightarrow X \) be the transformation that yields the values on system attributes obtained from a vector \( \gamma \in \Gamma_{Adm} \) of input parameters. We write, for \( \gamma \in \Gamma_{Adm} \), \( T(\gamma) = (T_1(\gamma), \ldots, T_n(\gamma)) \), with \( T_i: \Gamma_{Adm} \rightarrow X_i \). \( T_i(\gamma) \) allows observing the impacts of the specific configuration on the attributes. On the examples given above, the transformation \( T \) cannot be precisely known in a complex system. It requires complex simulations or experiments, and are thus costly and time consuming. Hence transformation \( T \) generally needs to be approximated by a qualitative model. Some examples of such qualitative models are given in Sections 4.1 and 8.

The attributes must then be interpreted in terms of satisfaction with regard to the company’s goals: a product rejection rate of 3% may be considered satisfactory for one company, whereas it remains intolerable for another, whose aim might be to achieve a “zero defect” production in order to obtain a standard certification. Moreover, all goals are not ascribed the same relative importance in company policy. This expression of preferences tends to be complex and requires an elaborate multi-criteria model relative to decision-making. Among these criteria, operational and monetary criteria are typically included. From the company’s standpoint, product performance is mandatory: low cost cannot compensate for poor operating performance. Consequently, the operational criteria act as a veto (a bad assessment on this criterion cannot be saved by good evaluations on the other criteria). Many other interactions among criteria, such as the conditional relative importance of criteria, are quite often encountered.

The interpretation of attributes with regard to company goals can be formalized through utility functions; also, the relative importance of goals or their preferential interactions between one another can be captured within the framework of Multi-Attribute Utility Theory (MAUT) [10, 11, 22]. MAUT consists of finding a real-valued utility function \( U \) such that for any pair of alternatives \( x, x' \), \( U(x) \geq U(x') \) if and only if alternative \( x \) is at least as good as \( x' \) for the decision maker relatively to all his concerns. When these alternatives are \( n \)-dimensional, i.e. \( X = (X_1, \ldots, X_n) \), the most widely used model
is the decomposable model derived by Krantz et al. [23], whereby $U$ assumes the form $U(x_1, \ldots, x_n) = F(u_1(x_1), \ldots, u_n(x_n))$, with $u_i : X_i \to [0, 1]$ being real-valued utility functions in $[0, 1]$ and $F : [0, 1]^n \to [0, 1]$ an aggregation function [19]. For $x_i \in X_i$, $u_i(x_i)$ is the extent to which the goal associated with criterion $i$ is satisfied by $x_i$. For $x \in X$, we set $u(x) = (u_1(x_1), \ldots, u_n(x_n)).$

All these concepts involving $\gamma, T, X, u, U, F$ are summarized in Figure 1 (Labels L1 to L5).

This formal model clearly distinguishes two points of view in the design of complex system improvements. Labels L1 to L3 capture the operational viewpoint and the behavioral description of a complex system, while labels L4 to L5 correspond to the strategic point of view and the preference model.

2.2 Description of the optimization problem

Given the existing system characterized by parameters value $\gamma^0 \in \Gamma_{Adm}$, the problem is to find an improved solution $\gamma \in \Gamma_{Adm}$ that reaches a minimum expected overall performance degree $U_{min}$ for $U(T(\gamma))$. Yet the company
will also consider the practical cost \( c(\gamma^0, \gamma) \) to improve the system \( \gamma^0 \) into \( \gamma \). Ultimately, the company would like to determine the solution that achieves an expected overall performance at the lowest cost.

\[
\begin{align*}
\min_{\gamma \in \Gamma_{Adm}} c(\gamma^0, \gamma) \quad \text{under} \\
\{ U(T(\gamma)) \geq U_{\min} \}
\end{align*}
\]  

The optimization problem (1) may be a very complex operation since we have seen that \( T \) is not explicitly known; moreover, it turns out to be very complex to perform a single computation of \( T \).

In order to reduce the number of calls of function \( T \), we propose to approximate \( T \) by a qualitative model. Such a qualitative model is widely used in engineering goal oriented model (e.g. [21, 15, 14]), industry design [6], qualitative Bayesian networks [32]. The qualitative model considered here concerns improvements. It will be described later (see Section 4).

Thanks to the qualitative model, we will perform successive improvements on the solution from the existing solution \( \gamma^0 \). In some domains, the user operates modification of the system directly by changing the parameters values \( \gamma \). But in other applications, the user does not have the hand directly on the parameters, but he can apply some improvement actions. We introduce thus the set \( A \) of system improvement actions. These actions can be direct individual modifications on the parameters. Examples of actions having more indirect impact of parameters are given in Section 7. Each action affects some change in some parameter(s) (e.g. increasing the value on a system parameter). Often, one needs to apply several actions to be able to improve significantly the overall performance.

Problem (1) is thus turned into an iterative approach in which at each iteration one aims at finding the set of actions which maximizes the improvement on the overall utility and minimizes the increase on the cost. Applying these actions, \( \gamma \) is changed. Then one computes \( U(T(\gamma)) \). The process continues until \( U(T(\gamma)) \) reaches minimal expected value \( U_{\min} \). This approach can be seen as some kind of steepest descent method.
Function GeneralProcess($A, U_{\text{min}}, \gamma$) :

\[ P \leftarrow (u_1(T_1(\gamma)), \ldots, u_n(T_n(\gamma))) \]
\[ p \leftarrow F(P) . \]

While ($p < U_{\text{min}}$) do

\[ B \leftarrow 2^N . \]

Repeat

\[ I^* \leftarrow \text{FindCoalition}(B, P) . \]
\[ S \leftarrow \text{FindActions}(A, I^*) . \]
\[ B \leftarrow B \setminus \{ I \subseteq N : I \supseteq I^* \} . \]

until ($S \neq \emptyset$)

The user selects one $ap \in S$ and applies it. This modifies $\gamma$.

\[ P \leftarrow (u_1(T_1(\gamma)), \ldots, u_n(T_n(\gamma))) . \]
\[ p \leftarrow F(P) . \]

done

return $\gamma$

End

Algorithm 1: General Algorithm GeneralProcess($A, U_{\text{min}}, \gamma$), where $A$ is the set of actions, $U_{\text{min}}$ is the minimal performance and $\gamma$ is the initial system parameters value. Initially, we call GeneralProcess($A, U_{\text{min}}, \gamma^0$).

The qualitative model actually directly relates each individual action to each criterion, in terms of the degree of belief that an improvement or a decrease over the criterion occurs. There is no direct information on the intensity of the increase/decrease that is expected. It is thus impossible to assess the impact on the overall evaluation. Hence the determination of the set of actions which maximizes the improvement on the overall utility and minimizes the increase on the cost is done in two separate steps:

- First, at the strategic level, we identify the criteria on which modifying their value offers the greatest reward. Given $\gamma \in \Gamma$, we first compute the vector $P = u(T(\gamma)) \in [0,1]^N$ providing the performance values on the $n$ criteria. Function FindCoalition($B, P$) determines the coalition $I^*$ of criteria belonging to the set $B$ of acceptable coalitions, for which the improvement of $P$ on all criteria in $I^*$ improves the overall evaluation in the most efficient way. We start with $B = 2^N$, $2^N$ being the power set of $N$. This takes into account both the aggregation function $F$ and the cost of the improvement.

- Second, at the operational level, one seeks for the set of actions which allows to improve all criteria in $I^*$ with the best expectation. Function FindActions($A, I^*$) returns several alternative sets of actions to be performed. This function does not return a single set of actions as
there are many potential sets which cannot be distinguished from our knowledge. The operator is asked to select the most appropriate one.

This two-step approach is described in Algorithm 1. Functions FindCoalition and FindActions are described in Sections 3 and 5 respectively.

In this two-step approach, it might occur that there is no set of actions that improves all criteria \( I^* \) identified at the first step. In this case, one needs to find another coalition \( I^* \). If it is not possible to improve all criteria in \( I^* \) at the same time, it will not be possible to improve all criteria in a superset of \( I^* \). Hence all supersets of \( I^* \) are discarded from the list of admissible set of criteria for \( B \).

3 Function FindCoalition: improvement recommendation at the strategic level

The MAUT framework merely captures designers’ preferences without any further considerations regarding material constraints beyond improvement implementation. Function FindCoalition(\( B, P \)) returns for a solution described by performance vector \( P \in [0,1]^n \) of scores w.r.t. criteria, the set \( I^* \) of criteria on which it is the more rewarding to improve the solution, independently of any operational constraints. The set \( I^* \) shall result from some kind of benefit to cost ratio analysis. On the one hand, the larger \( I^* \) the larger the benefit thanks to aggregation function \( F \). On the other hand, the larger \( I^* \) the larger the cost to perform an improvement on all criteria in \( I^* \). The difficulty here is that one does not know the intensity of the improvement actions will perform on criteria. Hence for each subset \( I \subseteq N \), an indicator that assesses the average benefit to cost ratio of improving criteria in \( I \) will be defined. Then \( I^* \) is defined as the subset \( I \) for which the associated indicator is maximal. Let us first consider when there is no cost of improvement.

An index (called worth index) denoted by \( \omega_I(F)(P) \) quantifying the worth for the vector \( P \) to be improved in criteria among \( I \subseteq N \), subject to the evaluation function \( F \), has been proposed in [25]. As exhibited in the following example, the subsets \( I \) are not to be restricted to single entries. Let us consider the case of a highly intolerant designer, as described by the \( \min \) aggregation function: \( F(P) = \min_{i \in N} P_i \). Should all the criteria be equally satisfied, then improving just one criterion will not alter the overall evaluation; hence, it is fruitless to work on a single criterion and instead worth improving all of the criteria simultaneously.

Let \( V \) be the set of piecewise continuous functions defined on \( [0,1]^n \). This space is endowed with the norm \( \forall H \in V, ||H||_V = \sup_{x \in [0,1]^n} |H(x)| \). The index \( \omega_I \) is considered as an operator from \( V \) into itself. The index \( \omega_I \) is
defined axiomatically for any $F \in V$. First of all, if $F$ is constant over criteria in $I$, then $\omega_{I}(F)(P) = 0$. Moreover, if $F$ is independent of criterion $i$, then $\omega_{\cup \{i\}}(F)(P) = \omega_{I}(F)(P)$. Another requirement is to provide a description, whenever $F$ can be decomposed into $n$ functions $F_{i}$ for each criterion, of an optimistic decomposability of $\omega_{I}(F)$ from the $\omega_{i}(F_{i})$. Lastly, an invariance property $\omega_{I}(F)$ for $\{0, 1\}$-valued functions $F$ is described. Previous requirements combined with linearity, symmetry and continuity (i.e. 

$$
\sup_{F \in V, F \neq 0} \frac{||\omega_{I}(F)||_{V}}{||F||_{V}} < \infty
$$

serve to uniquely define $\omega_{I}$ [25]:

$$
\omega_{I}(F)(P) = \int_{0}^{1} \left[ F((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I}) - F(P) \right] d\tau
$$

(2)

where $1_{I}$ is the identity function; and for every $d \in [0, 1]^{n}$, $(d_{I}, P_{N\setminus I})$ denotes a performance vector, with a component of $d$ on criteria in $I$, and the components of $P$ on the other criteria. With the notation $d_{I} = (1 - \tau)P_{I} + \tau 1_{I}$, the right hand side of (2) gives the mean value of the gain $F(d_{I}, P_{N\setminus I}) - F(P)$ only for improvement vectors $d_{I}$ on the segment from the current performance values $P_{I}$ (for $\tau = 0$) to the best possible improvement in $I$, i.e. $1_{I}$ (for $\tau = 1$). Equation (2) therefore yields the mean impact generated by uniformly improving all criteria in $I$ simultaneously, at which point it may be assumed that all possible levels of improvement (from sticking to $P_{I}$ until reaching the ideal vector $1_{I}$) have the same probability to occur. The diagonal from $P_{I}$ to $1_{I}$ is considered since it has been assumed that the improvements on all criteria are homogeneous.

Let us now introduce the cost in (2). The cost at the strategic level going from performance profile $P$ to $P'$ is denoted by $c_{str}(P, P')$. Note that this cost may be correlated with decisive factors that do not necessarily entail monetary considerations; they might actually be correlated with risk appraisal, temporal requirement, resources availability, etc. In such case, no relationship whatsoever exists between cost functions $c_{str}(P, P')$ and $c(\gamma, \gamma')$, and $c_{str}(P, P')$ captures an additionnal degree of freedom in the improvement design.

Relation (2) can be generalized by replacing the benefit factor by a benefit to cost ratio:

$$
\omega_{I}(F)(P) = \int_{0}^{1} \frac{[F((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I}) - F(P)]}{c_{str}(P, ((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I}))} d\tau
$$

(3)

Under the linear assumption $c_{str}(P, P') = \sum_{i \in N}(P'_{i} - P_{i}) uc_{str}^{i}$, with $uc_{str}^{i}$ a unit cost related to criterion $i$. Hence $c_{str}(P, ((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I})) = \tau \sum_{i \in I}(1 - P_{i}) uc_{str}$. When $c_{str}(P, P')$ is related to monetary considerations, value $c_{str}(P, ((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I}))$ will be considered as a mean cost for improving $P$ into $((1 - \tau)P_{I} + \tau 1_{I}, P_{N\setminus I})$, which is assessed independently of
any particular improvement action. This value can only be produced when
designer/operator’s know-how is available. For example, an experienced de-
signer/operator is assumed capable of assessing the cost related to employees
training as an average over years without precise knowledge of the allocated
training actions.

Finally, the subset $I^*$ of criteria that maximizes the worth index

$$\text{FindCoalition}(B, P) = \arg \max_{I \in B} \omega_I(F)(P) \quad (4)$$

indicates the mean performance to cost ratio that are the most beneficial to
be improved first.

4 Concepts for improvement recommendation at
the operational level

The operational constraints have been ignored during the identification of $I^*$. In function $\text{FindActions}(A, I^*)$, these operational constraints are considered, one aims at finding how the designer/operator should modify system $\gamma$ in order to improve criteria in $I^*$.

Before defining $\text{FindActions}(A, I^*)$, one needs to specify the qualitative influence model.

4.1 Goals-actions relationships

According to our approach, system improvements are carried out thanks to
system improvement actions. The set of actions is denoted by $A$. In order to
improve a subset of criteria, several actions generally need to be performed simultaneoulsy. There are some constraints among the actions so that some of them cannot be performed together: they are said mutually exclusive. An action plan is a set of actions that can be performed together. Let us begin by specifying the notion of action plan.

Definition 1 An action plan $ap$ is a subset of non exclusive actions, i.e. $ap \subseteq A$. The set of all action plans is denoted $AP \subseteq 2^A$.

In general, the set of actions and action plans can be defined independently of system parameters. However, actions can be defined directly as modifications of the system parameters.

Example 2 Let’s consider the case when two actions are related to each parameter $\gamma_j$ : this value can be increased or decreased. The two actions “$\gamma_j$ increase” and “$\gamma_j$ decrease” are mutually exclusive. Thus, $2p$ potential actions are available in $A$. Action plans are subsets of actions that concern different parameters.
We are interested in system improvements. Hence, instead of defining the complex transformation $T : \Gamma_{Adm} \rightarrow X$ that yields the values on system attributes obtained from a vector $\gamma \in \Gamma_{Adm}$, one asks the experts to provide the relationship between actions in $A$ and the goals (criteria) in a qualitative way as it is often the case in goal oriented models (e.g. [13, 15, 14]). We will use a fuzzy behavioral model. Indeed, as the gathered information originates from experts’ or managers’ own perception instead of being factually measured, it is intrinsically imprecise [36].

Our proposal is based upon the pioneering work of [6]. Felix’s fuzzy relationship model introduces two fuzzy subsets in order to distinguish actions with positive or negative impacts on performance [8].

**Definition 3** Consider a performance (criterion) $i \in N$ and an action $a \in A$. Let $S_i(a)$ be the degree of belief to which action $a$ can positively affect performance $i$. When $S_i(a) > 0$, we say that action $a$ satisfies $i$. Let $D_i(a)$ be the degree of belief to which action $a$ can negatively affect performance $i$. When $D_i(a) > 0$, we say that action $a$ distracts $i$. In other words $S_i$ (resp. $D_i$) can be seen as the fuzzy subset of actions which support (resp. distract) $i$. Action $a$ has no influence on $i$ when $S_i(a) = 0$ and $D_i(a) = 0$. The influence of an action is either positive, negative or null: $\min(S_i(a), D_i(a)) = 0$.

By considering the influence of actions over criteria directly without explicitly going through system parameters, we do not need to take into account constraints among parameters’ values.

From our perspective, this fuzzy model would seem to match the genuine expertise generally available from transforming $u \circ T$ into a complex system: the impact of an action on system performance can typically only be described in a qualitative and non-deterministic manner.

The fuzzy model described in Definition 3 can be represented through a digraph between $A$ and $N$ such that (see example in Figure 2): the arc between action $a$ and performance $i$ is equal to

$$\text{Influence}(a,i) = \begin{cases} +S_i(a) & \text{if } S_i(a) > 0 \\ -D_i(a) & \text{if } D_i(a) > 0 \end{cases} \quad (5)$$

4.2 Combining the influences of several actions: reasoning framework

The qualitative action-goal relationship needs to be extended to action plans. The major difficulty is that for an action plan $ap$ and a criterion $i$, several actions in $ap$ may affect $i$ positively and several other actions in $ap$ may affect $i$ negatively. Then what is the resulting effect of $ap$ on criterion $i$?

This problem could be solved by asking the experts to provide the degree of belief to which an action plan is supported to directly affect a criterion.
This is not realistic in practice due to the huge size of $AP$. The estimation of the merged impact of an action plan naturally depends on system behavior, as well as on the designer/operator’s decisional behavior: a pessimistic attitude (whereby a risk aversion position will focus attention to the most highly negative merged impacts of the action plan) vs. an optimistic attitude (whereby risk acceptance will focus attention on the most highly positive merged impacts). Hence the combination of positive and negative impacts cannot be solved objectively and is equivalent to a problem of decision under uncertainty.

Consider an action plan $ap \in AP$ and a criterion $i \in N$. We wish to define the resulting effect of $ap$ over $i$. This will be based on the actions $A_S^i(ap)$ (resp. $A_D^i(ap)$) that have a positive (resp. negative) effect on $i$:

$$A_S^i(ap) = \{a \in ap : S_i(a) > 0\} \quad \text{and} \quad A_D^i(ap) = \{a \in ap : D_i(a) > 0\}.$$  

(6)

The influence is basically bipolar as one can demarcate between positive influence (support) and negative one (denial). It is not wished to synthesize directly these contradictory stimuli. Hence we are in the bivariate context, where the resulting effect of $ap$ over $i$ is first assessed using two separate scales [17]: one for the positive stimuli in favor of supporting $i$, and one for the negative stimuli in favor of the denial.

**Definition 4** The bivariate scale is denoted by the pair $(S_i(ap), D_i(ap))$, where $S_i(ap)$ (resp. $D_i(ap)$) is the degree to which $ap$ satisfies (resp. denies) $i$.

This bivariate scale is for instance used in [14]. $S_i(ap)$ (resp. $D_i(ap)$) results from an aggregation of numbers $\{S_i(a), a \in A_S^i(ap)\}$ (resp. $\{D_i(a), a \in A_D^i(ap)\}$). The choice of the aggregation functions for the satisfaction/denial parts depends on the attitude of the designer/operator towards risk.

**Definition 5** Let us give two standard risk behaviors:

- Under Drastic Reasoning Framework (DRF), we have

$$S_i^{DRF}(ap) = \min_{a \in A_S^i(ap)} S_i(a) \quad \text{and} \quad D_i^{DRF}(ap) = \max_{a \in A_D^i(ap)} D_i(a).$$
According to DRF, the lowest belief in a positive impact value is to be compared to the highest belief in a negative impact value on \( i \). The designer/operator’s behavior represents a serious risk aversion and moreover describes a pessimistic attitude.

• Under Flexible Reasoning Framework (FRF), we have

\[
S_{i}^{\text{FRF}}(ap) = \max_{a \in A_{i}^{D}(ap)} S_{i}(a) \quad \text{and} \quad D_{i}^{\text{FRF}}(ap) = \max_{a \in A_{i}^{P}(ap)} D_{i}(a).
\]

According to FRF, the highest belief in a positive impact value is to be compared to the highest belief in a negative impact value on \( i \). In this instance, the designer/operator’s behavior more willingly accepts being exposed to the risk of committing an estimation error. This framework describes an optimistic attitude.

Other aggregation functions are of course possible, but we will stick to the previous two examples as they represent the most standard forms. Note that FRF is chosen in [14].

We need now to quantify the degree of belief to which an action plan is expected to improve a subset of performances in \( I^{*} \).

**Definition 6** Consider an action plan \( ap \in AP \) and a criterion \( i \). The resulting degree of belief to which \( ap \) should contribute to the improvement of \( i \) is:

\[
s_{i}(ap) = \begin{cases} S_{i}(ap) \text{ if } S_{i}(ap) > D_{i}(ap) \\ 0 \text{ else} \end{cases}
\]  

(7)

Note that condition “\( S_{i}(ap) > D_{i}(ap) \)” can be replaced by a more drastic condition such as “\( S_{i}(ap) > D_{i}(ap) + \eta \)” (where \( \eta > 0 \) is a threshold). Definition 6 concerns a single performance but can be extended to any subset of performances. We will define the degree of belief \( s_{I}(ap) \) to which \( ap \) should improve all criteria in \( I \). Two proposals are made.

**Definition 7** Consider a subset of performances \( I \subseteq N \) and an action plan \( ap \in AP \). The resulting degree of belief to which \( ap \) should contribute to the improvement of \( I \) is

\[
\tilde{s}_{I}(ap) = \min_{i \in I} s_{i}(ap)
\]  

(8)

The previous definition can be reinforced, by requiring no degradation on the remaining criteria on \( N \setminus I \).

**Definition 8** Consider a subset of performances \( I \subseteq N \) and an action plan \( ap \in AP \). The resulting degree of belief to which \( ap \) should contribute to the improvement of \( I \) while not deteriorating the other criteria is

\[
\hat{s}_{I}(ap) = \begin{cases} \min_{i \in I} s_{i}(ap) \text{ if } \forall j \in N \setminus I, [S_{j}(ap) > D_{j}(ap) \text{ or } A_{j}^{D}(ap) = \emptyset] \\ 0 \text{ else} \end{cases}
\]  

(9)
In other words, \( \hat{s}_I(ap) > 0 \) if any performance in \( I \) is improved by \( ap \), whereas no performance in \( N \setminus I \) is being deterred. For criterion \( j \) in \( N \setminus I \), note that the condition in (9) is fulfilled when \( A^P_j(ap) = A^S_j(ap) = \emptyset \).

Nevertheless, let us note that when the vector of performances is Pareto optimal, improving performances on criteria in \( I \) will necessarily give rise to a negative impact with regard to certain other performances, in which case the constraint in equation (9) cannot be satisfied. For example, in the DRF framework, a slight negative impact can be tolerated for performances in a subset \( I' \subseteq (N \setminus I) \); then, constraint [\( S_j(ap) > D_j(ap) \) or \( A^P_j(ap) = \emptyset \)] is maintained in equation (9) only for performances in \( N \setminus (I \cup I') \).

In the sequel, \( s_I(ap) \) will be equal either to \( \hat{s}_I(ap) \) or to \( \tilde{s}_I(ap) \).

**Definition 9** An action plan \( ap \in AP \) is admissible on \( I^* \) if \( s_{I^*}(ap) > 0 \). \( ap \) is admissible with degree \( \alpha \in [0, 1] \) if \( s_{I^*}(ap) \geq \alpha \).

**4.3 Aim of function** FindActions(\( A, I^* \))

We have just defined the degree to which an action plan satisfies a subset \( I^* \) of criteria. In parallel, an operational cost \( c_{op}(a) \) is associated with each action \( a \). The cost of action plan \( ap \) can thus be computed as follows: \( c_{op}(ap) = \sum_{a \in ap} c_{op}(a) \). In most instances, a cost constraint is applicable to the improvement project such that the cost of an action plan cannot exceed a predetermined budget.

Designers/operators focus on solutions with degrees of admissibility at least as high as a threshold \( \alpha_0 \) and whose cost does not exceed the budget constraint (budget threshold). Nevertheless, they would naturally prefer the solution with the highest degree of admissibility at an identical cost; conversely, they would prefer the least costly solutions should the degree of admissibility be the same. An order of preference on action plans is defined in order to capture this behavior. To develop this order, let’s consider the non-dominated Pareto action plans with respect to these two metrics: degree of admissibility \( s_{I^*}(ap) \) and cost \( c_{op}(ap) \). Since the preferences governing these two metrics lie in opposite directions, we will apply the Pareto order with respect to the couple \( (s_{I^*}(ap), -c_{op}(ap)) \). We make the following assumptions

\[
\alpha_0 > 0 \quad , \quad \text{budget} > 0 \quad , \quad \forall a \in A \quad c_{op}(a) > 0. \tag{10}
\]

Let us define the Pareto order \( \prec \) over action plans admissible of \( AP \) as:

\[
ap \prec ap' \iff (s_{I^*}(ap), -c_{op}(ap)) \prec_{\text{Pareto}} (s_{I^*}(ap'), -c_{op}(ap')) \tag{11}
\]

\[
\quad \iff [(s_{I^*}(ap) < s_{I^*}(ap')) \text{ and } (c_{op}(ap) \geq c_{op}(ap'))] \text{ or } [(s_{I^*}(ap) \leq s_{I^*}(ap')) \text{ and } (c_{op}(ap) > c_{op}(ap'))]
\]
Action plan $ap$ is dominated by action plan $ap'$ if $ap \prec ap'$. The search for efficient admissible action plans, with a cost lower than $budget$ and admissibility greater than $\alpha_0$ can then be stated as:

$$\max_{ap \in AP : s_i^*(ap) \geq \alpha_0 \text{ and } c_{op}(ap) \leq budget} (s_i^*(ap), -c_{op}(ap))$$

(12)

5 Function FindActions: improvement recommendation at the operational level

The objective of this section is to determine a set of admissible action plans at minimum cost and maximum degree of admissibility. In practice, the Pareto front of solutions for the problem in (12) is computed. This section will also present a Branch-and-Bound algorithm with appropriate heuristics in order to efficiently solve this problem. These heuristics depend on the adopted reasoning framework ($DRF$ or $FRF$ – see Def. 5).

Furthermore, several framework-dependent definitions are to be introduced in order to help explain algorithm heuristics.

Definition 10 Let $i \in N$

- **improvement**: For $i \in N \setminus I^*$, action plan $ap$ improves criterion $i$ when $A^D_i(ap) = \emptyset$ or $S_i(ap) > D_i(ap)$. For $i \in I^*$, $ap$ improves $i$ when $S_i(ap) > D_i(ap)$.

- **hindrance**: action plan $ap$ hinders criterion $i$ if $A^D_i(ap) \neq \emptyset$ and $S_i(ap) \leq D_i(ap)$.

- **compensation**: action $a$ compensates hindering actions in action plan $ap$ on criterion $i$ if $S_i(a) > \max_{a' \in A^D_i(ap)} D_i(a')$.

- **restriction**: an action $a$ is restrictive if $D_i(a) < \alpha_0$ for $i \in I^*$ and in case FRF, $a$ does not compensate for any hindering actions for performances in $N \setminus I^*$. An action $a$ is also restrictive if $c_{op}(a) > budget$.

5.1 The General Branch and Bound Principle

The Branch and Bound algorithm explores the set $AP$ of action plans. We assume here that there is no constraint among the individual actions so that $AP = 2^A$. Relaxing this assumption would require the use of Constraint Solving Problem (CSP) techniques. We do not consider this option to limit the size of the paper.

Introducing heuristics into the Branch-and-Bound algorithm allows selecting a relevant representation to efficiently derive solutions and cut irrelevant branches as quickly as possible.
Function $\text{SOLVE}(B, I^*, ap, S) : \rightarrow \text{Pareto-Front-of-Solutions}$

$S' \leftarrow S$;

If ( IS_ADMISSIBLE$(ap, I^*)$ ) then
   // 1: $ap$ is admissible – Add $ap$ to $S$ if non-dominated
   If ( $\not\exists ap' \in S / ap < ap'$ ) then
      $S' \leftarrow \{ ap \} \cup \{ ap' \in S / ap' \not\preceq ap \}$;
   end If
end If

If ( $\neg$IS_ADMISSIBLE$(ap, I^*) \lor$ framework uses FRF ) then
   // 2: Remove from $B$ the elements that yield non-admissibility
   $B' \leftarrow \text{REDUCE}(B, ap)$;
   If ( $B' \neq \emptyset$ ) then
      // 3: Select one element in $B'$
      $a \leftarrow \text{CHOOSE}(B', ap)$;
      // 4: Explore the selection of the other elements of $B'$
      $S'' \leftarrow \text{SOLVE}(B' \setminus \{ a \}, I^*, ap, S')$;
      // 5: Explore when $a$ is added to $ap$
      $S' \leftarrow \text{SOLVE}(B' \setminus \{ a \}, I^*, ap \cup \{ a \}, S'' )$;
   end If
end If
return $S'$;

End

Algorithm 2: Determination of the Pareto front of solutions $S$.

The general Branch-and-Bound algorithm (SOLVE) (see Algorithm 2) computes efficient, admissible action plans; it constructs and returns the Pareto front of solutions (Equation 12). The SOLVE function parameters are: $B$ (set of remaining actions, i.e. potential candidate actions not yet included in the action plan); $I^*$ the set of criteria to be improved first; $ap$ (the action plan under construction, such that $ap \cap B = \emptyset$); and $S$ (the Pareto front of previously known solutions). It is initially launched as \text{FindActions}$(A, I^*) = \text{SOLVE}(A, I^*, \emptyset, \emptyset)$.

According to Algorithm 2, the few lines just after 1 update the Pareto front of solutions when $ap$ is added. IS_ADMISSIBLE basically computes $(s_f(ap) \geq \alpha_0) \land (c_{ap}(ap) \leq \text{budget})$. In the algorithm, we also add to this condition the fact that FRF is used. Indeed, if $ap$ is not admissible, then under FRF, one can obtain an admissible action plan by adding some actions to $ap$. CHOOSE introduces heuristics to select and return the action to be chosen at the current node (Section 5.2). Lastly, REDUCE introduces heuristics to cut branches as quickly as possible by means of reducing the set of remaining candidate actions (Section 5.3).
5.2 Action Selection (CHOOSE)

The purpose of action selection \( \text{CHOOSE}(B, ap) \) is to enhance the search by choosing a relevant candidate action capable of leading to a conclusion (whether positive or negative) as quickly as possible. Heuristics are then used to select such an action with minimal computations. In DRF, the action selected is the one that generates the most drastic constraints so as to enable branch-cutting as quickly as possible. In FRF, supplementary heuristics are added related to the specificities of the model. An action selected with this heuristics hinders at least one criterion in \( N \), with as few as possible remaining actions capable of compensating for this hindrance. In both cases, the actions selected will severely constrain the next selections when added to \( ap \), hence reducing the search complexity of this branch. More precisely, two sets are introduced.

**Definition 11** For \( a \in B \), set \( \text{DIS}(a, ap, B) = \{i \in N : a \in A_i^D(B) \text{ and } D_i(a) > \max(S_i(ap), D_i(ap))\} \), where \( A_i^D(B) \) is defined in (6).

\( \text{DIS}(a, ap, B) \) is the set of criteria such that \( a \) acts negatively on \( i \) \( (a \in A_i^D(B)) \) with a degree greater than that involved in \( S_i(ap) \).

**Lemma 12** For every \( i \in \text{DIS}(a, ap, B) \), condition “\( S_i(ap \cup \{a\}) > D_i(ap \cup \{a\}) \)” does not hold.

**Proof:** Let \( i \in \text{DIS}(a, ap, B) \) and denote \( ap' = ap \cup \{a\} \). Firstly, as \( a \in A_i^D(B) \) then \( A_i^D(ap') \neq \emptyset \) (note that \( a \in A_i^D(ap') \)). Secondly, we have in both frameworks DRF and FRF: \( S_i(ap') = S_i(ap) < D_i(a) = \max_{b \in ap'} D_i(b) = D_i(ap') \). Hence “\( S_i(ap') > D_i(ap') \)” does not hold.

**Lemma 13** Let \( \beta \in ]0,1[ \), \( i \in N \), and set \( \text{IMP}(i, \beta, B, ap) = \{a \in B : a \in A_i^S(B), S_i(a) \geq \beta \text{ and } S_i(ap) < \beta\} \). Let \( a' \in B \) such that \( a' \in A_i^D(B) \), actions in \( \text{IMP}(i, D_i(a'), B, ap) \) can compensate hindering action \( a' \) when \( ap \) cannot guarantee the compensation.

**Proof:** Obvious.

The two previous sets in Lemmas 1 and 2 allow defining for \( B \) and \( ap \):

\( \text{Co}(B, ap) = \{a \in B : \exists i \in \text{DIS}(a, ap, B) \text{ s.t. } \text{IMP}(i, D_i(a), B, ap) \neq \emptyset\} \)

and (where \( MCo \) stands for Most Constraining selection)

\( \text{MCo}(B, ap) = \arg\min_{a \in B} \min_{i \in \text{DIS}(a, ap, B)} |\text{IMP}(i, D_i(a), B, ap)|. \)

Finally \( \text{CHOOSE}(B, ap) \) returns any element in \( \text{MCo}(B, ap) \).
5.3 Reducing the set of remaining actions (REDUCE)

REDUCE aims to remove ineffective actions from the remaining set of actions. Each time an action is deleted, the search complexity of the particular branch is halved. REDUCE is based on several elementary reductions that are iteratively applied; these reductions concern mutually exclusive, non-compensable, expensive, Pareto, or else are related to the cardinal of \( ap \). Some reductions may depend on the choice of the reasoning framework among \( DRF \) or \( FRF \), of synthesis function among \( \hat{s} \) and \( \hat{s} \). To keep the length of this paper reasonable, we will only present five elementary reductions.

5.3.1 Cardinality Reduction

**Lemma 14** Let \( ap \) be an action plan improving criteria \( I^* \). If \( |ap| > |N| \), then there exists \( ap' \subseteq ap \) with \( |ap'| \leq |N| \) such that \( ap < ap' \).

**Proof:** For every \( i \in \mathbb{N} \), the action \( a \in ap \) that realizes \( \max_{a \in ap} S_i(a) \) is denoted by \( a(i) \).

Let us define \( J \) as follows: when \( \hat{s} \) is used, set \( J = \{ i \in \mathbb{N} \mid I^* : A^S_i(ap) \neq \emptyset \} \); and when \( \tilde{s} \) is used, set \( J = \emptyset \). Let \( ap' = \{ a(i) : i \in I^* \cup J \} \). Clearly \( |ap'| \leq |N| \) and thus \( c_{op}(ap') < c_{op}(ap) \).

As \( ap \) is admissible then for every \( i \in I^* \cup J \), every \( a \in A^S_i(ap) \), and for propagation models \( DRF \) and \( FRF \):

- For \( DRF \): \( \min_{a \in A^S_i(ap')} S_i(a) \geq \min_{a \in A^S_i(ap)} S_i(a) > \max_{a \in A^D_i(ap)} D_i(a) \geq \max_{a \in A^D_i(ap')} D_i(a) \).
- For \( FRF \): \( \max_{a \in A^S_i(ap')} S_i(a) = \max_{a \in A^S_i(ap)} S_i(a) > \max_{a \in A^D_i(ap)} D_i(a) \geq \max_{a \in A^D_i(ap')} D_i(a) \).

When \( \hat{s} \) is used, if \( i \in \mathbb{N} \setminus (I^* \cup J) \), we have \( A^D_i(ap) = \emptyset \) since \( ap \) is admissible and \( A^D_i(ap') = \emptyset \). Hence \( s_{I^*}(ap') \geq s_{I^*}(ap) \) in propagation models \( DRF \) and \( FRF \).

This implies that branches with more than \( |N| \) elements are cut.

5.3.2 Locking Actions Reduction (LA)

Let us define \( LA(B, ap) \) as follows: when \( \hat{s} \) is used, we set

\[
LA(B, ap) = \{ a \in B : \exists i \in I^* \text{ s.t. } D_i(a) \geq \max_{a' \in B \cup ap} S_i(a') \}
\]

and when \( \tilde{s} \) is used, we set

\[
LA(B, ap) = \{ a \in B : \exists i \in \mathbb{N} \text{ s.t. } D_i(a) \geq \max_{a' \in B \cup ap} S_i(a') \}
\].
Each locking action in $LA(B, ap)$ would distract from performances in $N$ which could be compensated by any action in $B$ or $ap$. Indeed any action plan containing such an action will not be an admissible action plan due to relations (8) and (9). Hence the following result is shown.

**Lemma 15** There does not exist any $ap' \subseteq B \cup ap$ with $ap \subseteq ap'$ and $LA(B, ap) \cap ap' \neq \emptyset$ which is admissible.

### 5.3.3 Cost reduction (LC)

Let $LC(B, ap) = \{a \in B : c_{op}(ap \cup \{a\}) > \text{budget}\}$. None of the action in $LC(B, ap)$ can be added to $ap$ without exceeding the maximal allowed cost budget. Hence the following result is shown.

**Lemma 16** There does not exist any $ap' \subseteq B \cup ap$ with $ap \subseteq ap'$ and $LC(B, ap) \cap ap' \neq \emptyset$ which is admissible.

### 5.3.4 Incompatible Admissibility Reduction (INC)

This reduction only applies in framework $DRF$. Let $INC(B, ap) = \{a \in B : \exists i \in I^*, \exists a' \in ap \text{ s.t. } S_i(a') \leq D_i(a) \text{ or } D_i(a') \geq S_i(a)\}$.

**Lemma 17** There does not exist any $ap' \subseteq B \cup ap$ with $ap \subseteq ap'$ and $INC(B, ap) \cap ap' \neq \emptyset$ which is admissible.

**Proof** : Adding $a \in INC(B, ap)$ to $ap$ implies that there exists $i \in I^*$ such that

- in the case where there exists $a' \in ap$ s.t. $S_i(a') \leq D_i(a)$, then $S_i^{DRF}(ap \cap \{a\}) \leq S_i(a') \leq D_i(a) \leq D_i^{DRF}(ap \cap \{a\})$;

- in the case where there exists $a' \in ap$ s.t. $D_i(a') \geq S_i(a)$, then $S_i^{DRF}(ap \cap \{a\}) \leq S_i(a) \leq D_i(a') \leq D_i^{DRF}(ap \cap \{a\})$.

In both cases, $\tilde{s}_{1^*}(ap \cap \{a\}) = 0$ and $\tilde{s}_{1^*}(ap \cap \{a\}) = 0$. Under $DRF$, adding other actions cannot improve $s_{1^*}$.

### 5.3.5 Pareto Reduction (PAR)

Under DRF, we define $PAR_{DRF}(B, ap, S)$ by

$$\left\{ ap' \in S : \left( \min_{i \in I^*/A_i^+(ap) \neq \emptyset} S_i^{DRF}(ap) < s_{1^*}(ap') \right) \land \left( c(ap) \geq c(ap') \right) \right\} \lor \left( \min_{i \in I^*/A_i^+(ap) \neq \emptyset} S_i^{DRF}(ap) = s_{1^*}(ap') \right) \land \left( c(ap) > c(ap') \right)$$
Lemma 18 If $\text{PAR}_{\text{DRF}}(B, ap, S) \neq \emptyset$ then for all $ap'' \subseteq B \cup ap$ with $ap \subset ap''$ and $ap''$ is admissible, there exists $ap \in S$ such that $ap'' \prec ap'$.

Proof: Let $ap'' \subseteq B \cup ap$ with $ap \subset ap''$ and $ap''$ is admissible. As $\text{PAR}_{\text{DRF}}(B, ap, S) \neq \emptyset$ there exists $ap' \in S$ such that:

- $(s_{I^*}(ap'') \leq \min_{\{i \in I^*/A_i^*(ap) \neq \emptyset\}} S_i^{\text{DRF}}(ap) < s_{I^*}(ap')) \land (c(ap'') > c(ap) \geq c(ap'))$ then $(s_{I^*}(ap'') < s_{I^*}(ap')) \land (c(ap'') > c(ap'))$.
- Or $(s_{I^*}(ap'') \leq \min_{\{i \in I^*/A_i^*(ap) \neq \emptyset\}} S_i^{\text{DRF}}(ap) = s_{I^*}(ap')) \land (c(ap'') > c(ap))$ then $(s_{I^*}(ap'') \leq s_{I^*}(ap')) \land (c(ap'') > c(ap'))$.

In both cases, we have $ap'' \prec ap'$. ■

Thus, if $\text{PAR}_{\text{DRF}}(B, ap, S) \neq \emptyset$ adding any action in $B$ to $ap$ cannot yield a non Pareto dominated solution.

Pareto reduction in FRF requires a more complex partial order to define $\text{PAR}_{\text{FRF}}(B, ap, S)$ but the demonstration is similar. This heuristics is denoted $\text{PAR}(B, ap, S)$ when the framework is not specified.

5.3.6 REDUCE function

Finally, we have $\text{REDUCE}(B, ap) =$

\[
\begin{cases} 
  B \setminus (\text{LA}(B, ap) \cup \text{LC}(B, ap) \cup \text{INC}(B, ap)) & \text{if } (|ap| \leq |N| \land \text{PAR}(B, ap, S) = \emptyset) \\
  \emptyset & \text{else} 
\end{cases}
\]

5.3.7 Property of the algorithm

Theorem 19 Whatever the choice of the reasoning framework among DRF or FRF, and of synthesis function among $\hat{s}$ and $\tilde{s}$, Algorithm 2 terminates and returns a vector $S$ containing the Pareto non-dominated action plans in the sense of $\prec$.

Proof: The selected order of actions (function CHOOSE) exerts no influence on any of the algorithm’s termination, completeness and consistency properties.

Termination is obvious as we use a branch and bound algorithm and the size of the search space is finite. Thus we just have to show that the algorithm does not deter reaching a better action plan. To demonstrate the non deterioration, we just need to prove two proposals:

Assertion 20 Any action plan $ap$ minimal in the sense of the Pareto order on the action plans is necessarily an element of the set of solutions $S$. 

20
Proof: Let $ap$ be non-dominated. We have several cases in which the branch leading to $ap$ is cut:

- Assume that $ap$ is never reached because of condition IS\_ADMISSIBLE which cuts the branch that yields $ap$. This means that there exists $ap' \subset ap$ such that $ap'$ is admissible (i.e. IS\_ADMISSIBLE($ap'$, $I^*$) is true). We need to consider the two frameworks DRF and FRF:
  - Under DRF; as $ap'$ is admissible, we have $s_I^*(ap') \geq \alpha_0 > 0$. Hence as $ap' \subset ap$, $s_I^*(ap) \leq s_I^*(ap')$. This holds for both $\tilde{s}$ and $\hat{s}$. Moreover, $c_{op}(ap) > c_{op}(ap')$ (see (10)). Therefore
    $$(s_I^*(ap'), -c_{op}(ap')) \prec (s_I^*(ap), -c_{op}(ap)),$$
    which contradicts that $ap$ is Pareto non-dominated.
  - Under FRF, $ap$ cannot be blocked because of condition IS\_ADMISSIBLE does not prevent from exploring the sub-tree.

- Assume that we arrive at solution $ap$ which fulfills condition IS\_ADMISSIBLE, but in the instructions after $\textcircled{1}$, $ap$ is not kept. This implies that there exists $ap' \in S$ such that $ap \prec ap'$, which contradicts the fact that $ap$ is non-dominated.

- Assume that $ap$ is stored in $S$, but it is removed from $S$ later (see instruction after $\textcircled{1}$). This means that a new admissible action plan $ap'$ dominated $ap$ in the sense of $\prec$, which is again impossible.

- Assume that $ap$ is cut because of function REDUCE. We are after $\textcircled{2}$. According to Lemmas 15, 16 and 17, actions in LA($B, ap$), LC($B, ap$) and INC($B, ap$) can be removed from $B$ without changing the Pareto front of solutions. Moreover, when $|ap| \geq |N|$ or $\text{PAR}(B, ap, S) \neq \emptyset$, adding any action to $ap$ yields action plans that are necessarily dominated in the sense of $\prec$ (see Lemma 14 and Lemma 18) and cannot thus belong to the Pareto front of solutions. Hence the reductions in function REDUCE do not prevent from obtaining Pareto optimal solutions.

Finally, $ap$ is never cut unless it is stored in $S$. ■

Assertion 21 There is no element in $S$ that dominates another one in $S$ in the sense of $\prec$.

Proof: We have to prove that $S$ only contains elements which are not dominated by another element of $S$. Algorithm SOLVE is called recursively. We will prove this property by induction on the calls of algorithm
SOLVE. First of all, the first time the algorithm is called with $S = \emptyset$, which clearly satisfies the property. Hence we only need to show that if algorithm SOLVE($B, I^*, ap, S$) is called, where $S$ satisfies the property, then $S'$ that is returned also satisfies the property.

Let us consider a call SOLVE($B, I^*, ap, S$) where $S$ satisfies the property. Consider first the instructions just after $\textcircled{1}$. Then $ap$ is added to $S$ if there is no $ap' \in S$ such that $ap \prec ap'$. In this case, all elements in $S$ that are dominated by $ap$ are removed. Then $S'$ is either set to $S$ or set to $(S \cup \{ap\}) \setminus \{ap' \in S : ap' \prec ap\}$. Thus $S'$ also satisfies the property.

$S'$ can also be changed in instructions after $\textcircled{4}$. By the induction assumption, as $S'$ satisfies the property, then $S''$ also satisfies the property. We repeat this reasoning for the instruction after $\textcircled{5}$. Hence, at the end, $S'$ satisfies the property.

Combining Assertions 20 and 21, we obtain that the result of the algorithm contains exactly all non-dominated action plans in the sense of $\prec$.

6 Experiments related to FindActions function efficiency

Experimentations have been carried out upon a set of about 156000 automatically generated problems. Each problem is defined by a digraph structure. $|A|$ ranges from 10 to 200, and $|N|$ from 5 to 30 in the benchmark domain. Let denote $NNA = \{(a, i) \in A \times N : max(S_i(a), D_i(a)) > 0\}$ (arcs with non null effect) and $NNAP = \{(a, i) \in A \times N : S_i(a) > 0\}$ (arcs with positive effect). A program has been developped to randomly generate digraphs whose parameters are: $|A|$, $|N|$, the percentage of $NNA$, the percentage of $NNAP$ among $NNA$. Finally, several values of $\alpha_0$ and allocated budget have also been included in the tests. The range values of all these parameters are provided in Table 1. The cost of an action $a$ has been defined as the number of positive arcs that connect it to performances. A minimal influence degree threshold has been set to 0.2.

<table>
<thead>
<tr>
<th>Digraphs domain</th>
<th>Resolution parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>A</td>
</tr>
<tr>
<td>10 to 200 (Step by 5)</td>
<td>5 to 30 (Step by 5)</td>
</tr>
</tbody>
</table>

For each $|A|$, $|N|$, $%|NNA|$, about 100 digraphs are randomly generated. Then, $%|NNAP|$ is randomly associated to each digraph.

Table 1: Domain of digraphs covered and resolution parameters
Each generated digraph has then been solved for both the DRF and FRF. Note that the aim of these experimentations is to test the algorithm’s efficiency instead of the implementation optimization which is out of concern of this paper (see perspectives of the section). Consequently, the evaluation of the complexity of the resolution has been done by counting the number of times the CHOOSE method is called (number of explored nodes) instead of execution time which depends on implementation optimization. This allows to get an estimation of the speed of the algorithm which is independent from the CPU speed. Let us denote this number $ntc$, it is to be compared to a maximum of $2^{|A|+1}$ which is the complexity of the problem to be solved (see section 5.1). Let also denote for a given value $limit$: $uns(limit)$ the number of unsolved problems\footnote{Problems for which the algorithm neither proves their inconsistency nor finds the whole Pareto front of solutions after $ntc$ calls to CHOOSE.} after at most $limit$ calls of the CHOOSE method, and $ns(limit)$ the number of problems among $uns(limit)$ where no solution (i.e. $\alpha_0$–admissible action plan with cost lower than budget) has been found after $limit$ calls of the CHOOSE method.

**Remark 22** On a single CPU, $ntc = 10^6$ explorations are performed in less than 10 minutes for the domain of problems which is covered by the test.

First we observe that the algorithm while searching for the Pareto front of solutions gives goods results especially for DRF. Indeed, 85% of the digraphs are solved with $ntc < 10^3$, and 97.4% are done with $ntc < 10^6$ (i.e., only 4055 problems among 156000 remain unsolved: $uns(10^6) = 4055$). Finally, only 3291 (2.1%) exceed the limit $2 \times 10^6$: $uns(2 \times 10^6) = 3291$ (see Table 2). The results are less relevant with FRF but the algorithm remains efficient: 66% are solved with $ntc < 10^3$ and only 2957 among 70651 (4.2%) problems exceed the limit $2 \times 10^6$: $uns(2 \times 10^6) = 2957$. Relaxation of constraints due to the FRF specificities mainly explained these results. This relaxation makes preliminary reductions inefficient and other reductions arise late in the search tree (see Table 3 for more details concerning the preliminary reductions).

<table>
<thead>
<tr>
<th>Nb Exp</th>
<th>limit</th>
<th>$0$</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
<th>$10^6$</th>
<th>$2 \times 10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRF</td>
<td>156000</td>
<td>$uns$</td>
<td>49848</td>
<td>31682</td>
<td>23439</td>
<td>14818</td>
<td>7079</td>
<td>4055</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ns$</td>
<td>49848</td>
<td>3056</td>
<td>1590</td>
<td>930</td>
<td>607</td>
<td>435</td>
</tr>
<tr>
<td>FRF</td>
<td>70651</td>
<td>$uns$</td>
<td>39713</td>
<td>31593</td>
<td>23889</td>
<td>15336</td>
<td>8340</td>
<td>3781</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ns$</td>
<td>39713</td>
<td>12522</td>
<td>10129</td>
<td>6724</td>
<td>4011</td>
<td>2222</td>
</tr>
</tbody>
</table>

**Table 2:** Pareto Resolution and Find One Solution Speed

If we only focus in finding one solution\footnote{The algorithm stops when the first admissible solution is found} instead of searching for the Pareto front, the results are much better for both frameworks (See Table 2). We can notice that for DRF, only 22 among 156000 (0.015%) problems exceed the limit $2 \times 10^6$: $ns(2 \times 10^6) = 22$ and 268 among 70651 (0.3%) for FRF.
This also means that the number of problems without solution that remain unsolved is very low since these values (22 resp. 268) are the upper limits respectively for \textit{DRF} (resp. \textit{FRF}). Thus most of the remaining unsolved problems have solutions (more than 99.3% resp. 90%). In the following we will focus on those ones.

In figures 3 and 4 several functions are plotted: \textbf{choose} is \textit{mean}(ntc), \textbf{sol}, \textbf{min} and \textbf{max} are respectively \textit{mean}, \textit{min} and \textit{max} (number of explored solutions), \textbf{comb} is \textit{mean}(\sum_{i=0}^{\min(|N|,|B|)} |B|^i) (where \sum_{i=0}^{\min(|N|,|B|)} |B|^i) is the upper bound of ntc taking into account that the cardinality of an action plan shall not exceed $|N|$ according to Cardinality reduction – See Section 5.3.1), \textbf{red} is \textit{mean}(number of reductions) and \textbf{canonic} is \textit{mean}(2^{|B|+1}). These figures use logarithmic scale (base 2). Figure 3 shows that \textbf{choose} and \textbf{sol} have approximately the same behaviour. By examining them in more details, we find a polynomial relation between them such that \textbf{choose} $\leq (\textbf{sol})^k$ (See Figure 5). The average value for $k$ is 1.8 (resp. 1.9) for \textit{DRF} (resp. \textit{FRF}).

![Figure 3: Behaviour of choose, sol, min, max, comb, red and canonic related to number of actions for DRF and FRF.](image)

**Remark 23** We limit the number of performances $|N|$ to a maximum of 30 because, as shown in Figure 4, the average complexity (choose) of problems becomes lower when the number of performances increases.

Now, let us examine in more details the efficiency of the algorithm and its heuristics. We can distinguish three kinds of reductions:

- canonical or preliminary reductions: these preliminary reductions (\textit{Locking Actions} and \textit{Restricting Actions} reductions) are performed before any exploration of the search space and allow reducing the problem into
Figure 4: Behaviour of choose, sol, min, max, comb, red and canonic related to number of performances for DRF and FRF.

Figure 5: Relation between number of choose and number of found solutions

its canonical form which depends on $\alpha_0$. Several $\alpha_0$ have been tested for both frameworks and in each case the mean number of eliminated actions has a linear relation with the initial number of actions. Table 3 shows the slope of the linear relation for each $\alpha_0$ (mean percentage of eliminated actions). These reductions may be very significant: for example in DRF, a mean of 63% of the actions are eliminated for $\alpha_0 = 0.3$. This suppresses $2^{|A|} - 2^{0.375|A|}$ nodes (about $1.6 \times 10^{60}$ for $|A| = 200$ with a canonical problem size reduced to about $1.9 \times 10^{22}$).

For this reason, the efficiency of heuristics applied after this step are to be related to the size of the canonical problem (see figures 3 and 4).

- structural reductions: the only one is Cardinality Reductions. It is only linked to the size of the problem ($|A|$ and $|N|$). It will reduce
The values obtained for \( \alpha_0 = 0.2 \) apply for any \( \alpha_0 < 0.2 \) as it is the minimal influence degree of each positive arc of the experiments.

Table 3: Slope of lines

<table>
<thead>
<tr>
<th>( \alpha_0 )</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRF</td>
<td>0.28</td>
<td>0.63</td>
<td>0.77</td>
<td>0.85</td>
<td>0.92</td>
<td>0.93</td>
<td>0.96</td>
<td>0.98</td>
</tr>
<tr>
<td>FRF</td>
<td>0.29</td>
<td>0.31</td>
<td>0.35</td>
<td>0.39</td>
<td>0.48</td>
<td>0.52</td>
<td>0.63</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Figure 6: Mean number of reductions w.r.t \(|A|\)

the amount of the search space to be explored to

\[
\min(|N|,|A|) \sum_{i=0}^{\min(|N|,|A|)} \binom{|A|}{i}
\]

instead of \(2^{|A|}\). **Cardinality Reductions** are applied during the exploration of the search space and limit the exploration depth. In fact, most of the time they are not needed during the search because other reductions have already cut the branches before reaching \(N\). Figure 6 shows its impact on the reduction of the search space: it provides the mean number of reductions as a function of \(|A|\).

- **semantic reductions**: They consist in **Cost**, **Locking Actions**, **Incompatible Actions** and **Pareto** reductions. Figure 6 shows their impact on reducing the search space.

At first sight, **Incompatible Actions** and **Locking Actions** seem the main contributors to the reductions. In fact, they act as the garbage collector of the algorithm: when complex deductions have suppressed or chosen an action, then they reduce the problem. Thus reductions are mostly attributed to them whereas they are not the genuine initiators. In addition **Cost**, **Cardinality** and **Pareto** reductions occur later during the exploration which limits their contributions.
6.1 Experiments summary

About 100 days of equivalent single CPU computing time have been required by the experiments described in this section (30 real days on 3 CPUs). Experiments have covered all (resp. half) the digraphs domain of experimentation for DRF (resp. FRF) in accordance to parameters' variations in Table 1. 156000 digraphs for DRF and 70651 for FRF, i.e. more than 225000 have been solved. These experiments have emphasized the efficiency of the algorithm. Indeed, it did not manage to find at least one solution after $2 \times 10^6$ iterations ($ntc$) for only $(22+268) = 1.28 \times 10^{-3}$. Furthermore, for most of these problems the whole set of solutions (the Pareto front) was found in less than $2 \times 10^6$ iterations. This is to be compared to the real complexity of the problem which initially ranges from $2^{50} \approx 10^{15}$ for the smallest problems to $2^{200} \approx 10^{60}$ for the biggest ones.

Several directions may be envisaged to further enhance the algorithm’s performances and are explicited in the final conclusion.

7 Case Study

Our case study concerns the adaptive management of a manufacturing plant. Strategic improvement priorities evolve over time, depending on the competitive context, and new improvement actions must be continuously planned to compensate contextual disturbances. This illustrative example does not claim that the decisions proposed by the algorithm are relevant from an industrial engineering point of view. For example, certain constraint relationships between actions are concealed in the model because they would require further knowledge regarding industrial improvement methods.

This case study is presented from a managerial viewpoint rather than from an industrial engineering perspective. For this reason, the improvement actions specified in this case study may indifferently be either improvement methods (e.g. Kanban) or strategic schedules (e.g. staff training). This ambiguity could be criticized from a purely industrial engineering point of view, though a full understanding of this illustrative example does not require any further attention.

The company’s overall objective is to increase performance, as measured in terms of customer satisfaction. On the one hand, four criteria are established by the company to assess its overall performance: $RP$ : Product Range, $PP$ : Product Pricing, $PQ$ : Product Quality, and $TD$ : Delivery Time; these are then complemented by an internal criterion: $SC$ : Social Climate. On the other hand, actions involved in the relationships with such performances are to be envisaged; these may correspond to implementing industrial performance improvement methods. At first, six classical improvement methods will be examined: Kanban (Kanban is related to lean and just-in-time production), SMED (Single-Minute Exchange of Die provides a
rapid and efficient approach to converting a manufacturing process from producing the current product to producing the next product), *Poka-Yoke* [26] (denoted POK - a Poka-Yoke is any mechanism within a lean manufacturing process that helps an equipment operator avoid mistakes; its purpose is to eliminate product defects by preventing, correcting or signaling human errors as they occur), *Six Sigma* (denoted 6Σ - Six Sigma seeks to improve the quality of process outputs by identifying and removing the causes of defects and minimizing variability in manufacturing and business processes), *TQM* (Total Quality Management is a set of management practices introduced throughout the organization and geared to ensuring that the organization consistently meets or exceeds customer requirements), and *TPM* (Total Predictive Maintenance is a new way of looking at maintenance, according to a proactive approach that is essentially aimed at preventing any kind of slack before it occurs). Strategic schedules can then be envisaged: *SRM* (supplier relationship management), *PR* (partial relocation in search of less expensive labor), *ST* (staff training), *TP* (traceability policy), and *CNP* (respect of standards and conformance with new policies).

These six industrial improvement methods and five strategic schedules define the set of actions ($|A| = 11$) in our framework; they may be combined to design action plans, as proposed in Section 4 to improve the industrial system. Moreover, we assume that they may be carried out independently of one another to improve performance.

In our framework, it is supposed that each improvement method or strategic schedule may be associated with an industrial system parameter but it is not necessary to make this connection explicit.

Table 4 provides the influence $\text{Influence}(a,i)$ of actions over criteria (see (5)). As an example, action *PR* should very probably have a positive impact on *PP*, whereas it should clearly have a negative impact on *SC* inside the company.

Let us suppose the company’s overall performance is defined as the aggregation $F$ of the five criteria with a Choquet integral [18]. The family of Choquet Integrals provides aggregation functions that enable accommodating both the relative weighting of criteria and their interactions; it covers a wide range of preference models. In this application, we consider a particular case of Choquet integrals, based on the so-called 2-additive measure [20, 16]: according to this simplified model, only interactions by pairs of criteria are considered. The 2- additive Choquet Integral can then be expressed, for an element $(y_1, \ldots, y_n) \in \mathbb{R}^n$ interpretable form as follows [27]:

$$\text{CI}(y_1, \ldots, y_n) = \sum_{i=1}^{n} \nu_i y_i - \frac{1}{2} \sum_{i>j} I_{ij} |y_i - y_j|$$

with the property that $((\nu_i - \frac{1}{2} \sum_{i\neq j} |I_{ij}|) \geq 0)$ and where the $\nu_i$ are the Shapley indices, representing the importance of each criterion relative to all the
Improvement method | SC | TD | PP | RP | PQ |
---|---|---|---|---|---|
Kanban | −0.5 | +0.5 | +0.9 | +0.6 | +0.2 |
SMED | −0.5 | +0.8 | 0. | +0.7 | +0.3 |
POK | +0.5 | +0.2 | −0.3 | −0.4 | +0.6 |
$\delta \Sigma$ | −0.2 | +0.5 | −0.2 | 0. | +0.8 |
TQM | +0.1 | −0.3 | −0.2 | 0. | +0.9 |
TPM | 0. | +0.8 | +0.3 | +0.4 | 0. |
SRM | +0.6 | +0.7 | −0.4 | +0.7 | 0. |
PR | −0.8 | 0. | +0.9 | 0. | −0.5 |
ST | +0.9 | 0. | −0.6 | +0.5 | +0.7 |
TP | +0.3 | +0.2 | −0.2 | −0.3 | +0.8 |
CNP | 0. | −0.3 | −0.5 | −0.4 | +0.8 |

Table 4: Influence matrix

others, with $\sum_{j=1}^{n} \nu_j = 1$; $I_{ij}$ denotes the interactions between pairs of criteria $(i, j)$ with values contained in the interval $[-1, 1]$. A value 1 signifies full complementarity between the two criteria (which are expected to be simultaneously satisfied), a value of −1 indicates full redundancy, and a null value means that the criteria are independent.

Table 5 lists the company’s initial performance $P$ vector for $\gamma^0$, along with the Shapley index of each criterion. The approximate average cost required to improve performance from 0 to 1 ($u_{str}^i$) for each criterion is also provided in Table 5. The interactions between criteria are given in Table 6. In the context relative to $P(\gamma^0)$, the preference model embedded in the Choquet integral could be synthesized with a rule of the following type: the company pays special attention to quality and product pricing policies, yet these policies must be accompanied by adequate services relative to the product range and delivery time in order to be truly attractive (positive interactions); moreover, the company must preserve its social climate (independence). In this illustrative example, we assume that all actions yield the same cost, say $c_{op}(a) = 1$ for all $a \in A$. We assume that the expectation of the decision maker is $U_{\text{min}} = 0.85$. Moreover $\alpha_0 = 0.2$ and budget $= 5$.

<table>
<thead>
<tr>
<th>Indicators</th>
<th>Shapley Index($\nu_i$)</th>
<th>$u_{str}^i$</th>
<th>$P^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Social Climate (SC)</td>
<td>0.10</td>
<td>1000k€</td>
<td>0.7</td>
</tr>
<tr>
<td>2 Delivery Time (TD)</td>
<td>0.10</td>
<td>3000k€</td>
<td>0.8</td>
</tr>
<tr>
<td>3 Product Pricing (PP)</td>
<td>0.25</td>
<td>4000k€</td>
<td>0.4</td>
</tr>
<tr>
<td>4 Product Range (RP)</td>
<td>0.05</td>
<td>2000k€</td>
<td>0.7</td>
</tr>
<tr>
<td>5 Product Quality (PQ)</td>
<td>0.50</td>
<td>4000k€</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 5: Weights, costs and initial performances
Let us look at the first two iterations of Algorithm 1.

**Iteration 1** starting from $\gamma^0$:

- $P^1 = (0.7, 0.8, 0.4, 0.7, 0.4)$ and thus $p^1 = CI(P^1) = 0.454$. As $p^1 < U_{\text{min}}$, we need to find suitable action plan to improve $\gamma^0$.
- **FindCoalition**$(2^N, P^1)$: For profile $P^1$, the coalition on which $\omega_I(CI)(P^1)$ is the largest is $\{PP, PQ\}$ (cf. Table 7). This result is completely natural as these two criteria have the worst scores and also have the largest importance. Moreover, there is a strong positive interaction among them. Hence, it is more rewarding to improve both $PP$ and $PQ$ rather than any of them individually ($\omega_{\{PP\}}(CI)(P^1) = 1.058E - 5$ and $\omega_{\{PQ\}}(CI)(P^1) = 7.967E - 5$). The recommendation to improve $\{PP, PQ\}$ is just ahead that for $SC$ alone because of the relative costs of improving $PP$ and $PQ$.

<table>
<thead>
<tr>
<th>Improvement method</th>
<th>$SC$</th>
<th>$TD$</th>
<th>$PP$</th>
<th>$RP$</th>
<th>$PQ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SC$</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>$TD$</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>$PP$</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.05</td>
<td>0.45</td>
</tr>
<tr>
<td>$RP$</td>
<td>0.</td>
<td>0.05</td>
<td>0.05</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>$PQ$</td>
<td>0.</td>
<td>0.1</td>
<td>0.45</td>
<td>0.</td>
<td>0.</td>
</tr>
</tbody>
</table>

Table 6: Interaction coefficients

- **FindActions**$(A, I^*)$ with $I^* = \{PP, PQ\}$. The result of algorithm SOLVE (Algo. 2) are presented in Table 8 for all possible choices among $DRF$ vs. $FRF$ and $\hat{s}$ vs. $\hat{s}$. Assume that the decision maker chooses to implement $TQM, PR$, which is supposed to have improvement impact on $PP, PQ$. but degrade a little bit the performance on $SC, TD$.

**Iteration 2** starting from $\gamma^1$:

- $P^2 = (0.6, 0.7, 0.8, 0.7, 0.8)$ and thus $p^2 = CI(P^2) = 0.7575$. As $p^2 < U_{\text{min}}$, we need to find suitable action plan to improve $\gamma^1$.
- **FindCoalition**$(2^N, P^2)$: $\omega_I(CI)(P^2)$ is maximum for $SC$ (see Table 9).
Table 8: Pareto front for all possible choices among DRF/FRF and \( \tilde{s}_I \)\((ap) \) and \( \tilde{s}_I \)\((ap) \). For each Pareto set, we show in bracket the vector \((s_I \)\((ap), -c_{op}(ap))\).

<table>
<thead>
<tr>
<th>( s_I )((ap) )</th>
<th>DRF</th>
<th>FRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>{TQM,PR} (0.9,-2)</td>
<td>{TQM,PR} (0.9,-2)</td>
<td>{TQM,PR} (0.9,-2)</td>
</tr>
<tr>
<td>{Kanban,TQM} (0.9,-2)</td>
<td>{Kanban,TQM} (0.9,-2)</td>
<td>{Kanban,TQM} (0.9,-2)</td>
</tr>
<tr>
<td>{PR,ST} (0.7,-2)</td>
<td>{PR,ST} (0.7,-2)</td>
<td>{PR,ST} (0.7,-2)</td>
</tr>
<tr>
<td>{Kanban,ST} (0.7,-2)</td>
<td>{Kanban,ST} (0.7,-2)</td>
<td>{Kanban,ST} (0.7,-2)</td>
</tr>
<tr>
<td>{Kanban,TQM,SRM} (0.9,-3)</td>
<td>{Kanban,TQM,SRM} (0.9,-3)</td>
<td>{Kanban,TQM,SRM} (0.9,-3)</td>
</tr>
<tr>
<td>{Kanban,TQM,ST} (0.9,-3)</td>
<td>{Kanban,TQM,ST} (0.9,-3)</td>
<td>{Kanban,TQM,ST} (0.9,-3)</td>
</tr>
</tbody>
</table>

Table 9: \( \omega_I(CI)(P^2) \) for several coalitions \( I \).

<table>
<thead>
<tr>
<th>( I )</th>
<th>( \omega_I(CI)(P^2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{SC}</td>
<td>(1E-4)</td>
</tr>
<tr>
<td>{SC,PP,PQ}</td>
<td>(8.75E-5)</td>
</tr>
<tr>
<td>{PP,PQ}</td>
<td>(8.4375E-5)</td>
</tr>
</tbody>
</table>

- FindActions\((A,I) \) with \( I = \{SC\} \). This time, the decision maker decides to use DRF and \( \hat{s} \) (he does not want any performance decrease on the criteria). The application of algorithm SOLVE gives Kanban,ST (with \(0.9,-2\)) and PR,ST (with \(0.9,-2\)).

Once the improvement becomes more complex, i.e. greater number of performances to be improved simultaneously, it becomes obvious that the Optimistic Attitude offers a broader range of admissible solutions. Furthermore, these solutions are considered with higher degrees of admissibility in the latter case.

8 Related Works

As described in Section 2, two challenges appear when designing a system improvement: at the strategic level, which changes to system outputs would be expected to bring significant improvements?; and at the operational level, which system parameter adjustments should be carried out in order to achieve the expected improvement?

In the Industrial Engineering literature, industrial performance management provides a considerable body of work relying on preference models. A Performance Measurement System (PMS) consists of a multi-criteria instrument for informing and supporting decision-makers in the activities of diagnosis and improvement [4, 31, 24]. Given its nature, a PMS thus requires the use of multi-criteria methods [35]. It solely focuses on strategic
management considerations and goal-based reasoning. For this reason, an alternative point view can be found in the Industrial Engineering literature. Felix criticized the way preference models were established for the purpose of designing improvement projects [6]. According to his mindset, a preference model must capture the cooperative or competitive nature of goals.

Another thorny issue encountered when designing a complex system improvement pertains to the transformation that provides expected improvements relative to the system goals obtained from an input parameter vector.

The qualitative model describing the relationships between input parameters and goals can be represented by a causal digraph [28]. A vast literature exists on the use of causal digraphs to describe oriented relationships between inputs (e.g. input parameters) and goals within complex systems [9]. These relationships constitute relevant models for both cognitive and explanatory purposes. Cognitive Maps (CM) offer one such trend [33]. Because of their user-friendly semantic, qualitative causal digraphs appear to be appropriate models for capturing the knowledge available pertaining to the complex input parameters-to-goals transformation in industrial settings.

The general action concept may be preferred to the parameter adjustment concept in the literature. The influence associated with an arc then corresponds to the notion that actions either support or hinder the achievement of a goal. The purely qualitative representation of action-goal relationship is proposed in [28]. For cases when the influence of actions can be more accurately characterized, a fuzzy relationship model has been proposed by Felix [6, 7].

The next issue to be addressed pertains to how the influences of actions on a specific goal are combined in an imprecise and/or uncertain context. Works completed relative to this issue can be found in Requirements Engineering [26] and Software Engineering [15, 14]. In these goal-driven models, the relation derived between an action \(a\) and a goal \(i\) may be expressed by degrees of satisfaction and denial for a given goal may be chosen in a qualitative setting [5, 14, 15]. When several actions for goal \(i\) exist in the digraph, the overall degree of satisfaction (resp. denial) for \(i\) is equivalent to the conjunction of degrees of satisfaction (resp. the disjunction of degrees of denial proof) of the goal’s previous actions in the digraph. This notion is close to those presented in [29, 28], whose authors combined this goal-driven representation with Felix’s goal relationship model [6, 7].

Other works in the field of Model-driven Engineering also require a model of the qualitative impacts between actions on parameters and system goals, e.g. the Dynamic Adaptive Systems (DAS) community [13]. Since many systems need to be adapted to an evolving context, Model-driven Engineering approaches are aimed at describing models relative to both their design time and runtime, thus providing the possible DAS configurations. In [12, 13], a Domain-Specific Modeling Language (DSML) has been proposed to represent variability in a system. Such language is more general than our represen-
tation of all potential actions. The main objective in all these works can be summarized as the search for an adequate parameter configuration that improves the level of satisfaction of a set of goals, possibly prioritized or interacting, whenever the known relationships between actions and goals are imprecise impacts or uncertain influences.

Our approach differentiates from the existing ones in the following way. First of all, it may be confusing to associate preferential interactions between goals and the behavioral influences between actions and goals from a semantic point of view. This approach may result in misleading interpretations as regards the improvement project. Secondly, it may be simpler in practice to break down the improvement process into steps and then distinguish preferential goal-based interactions and action-goal relationships when the system is overly complex. The formal model presented in Section 2 illustrates this “will vs. act” breakdown and supports the point of view being defended in this paper.

9 Conclusion

This paper has been devoted to proposing a trade-off between managerial and implementation aspects of industrial improvements. The formal model detailed in Section 2 distinguishes which choices during the design of an improvement project depend on the operational constraints of the required system from those choices related to the designer/operator’s strategic preferences. Based on this framework, the other sections have been dedicated to mathematically supporting the search for an efficient improvement, as required in a complex system.

The MAUT model enables synthesizing managerial preferences with respect to the performances to be improved first. Managerial preferences have been recorded in an analytical form that facilitates the search for strategic improvements relative to optimization problems, which therefore provides a powerful artifact for recording overall company performance and deriving a rationale from a managerial perspective. The MAUT model thus supports identifying the way in which a system should appropriately evolve in order to improve its performance to the greatest extent possible. The preference model is then complemented by other models that take into account the operational context. More specifically, a model of relationships existing between the system’s performance and its input parameters is needed to successfully complete the improvement implementation component. In this aim, our paper has proposed a fuzzy model in Section 4; the fuzzy digraph not only captures the system’s behavioral constraints, but it also includes the step of interpreting system attributes in terms of performance. It must not be considered therefore as a classical behavioral model of a system since its outputs depend on the designer/operator’s risk adversity and his optimistic vs.
pessimistic behavior regarding the impacts of actions he intends to carry out in order to achieve the expected performance. Subjectivity has thus been introduced into the behavioral model.

It has also been explained herein how both models must be used in an iterative procedure to design an efficient system improvement: qualitative knowledge about the system prevents assessing the precise impacts of improvement actions in a deterministic manner. The iterative improvement process is thus supported by a mathematical model, in addition to a software tool that allows testing our approach on an industrial case study.

In conclusion, the way in which these models are jointly used within our entire design procedure has been intended to prove that both models should be used in tandem in order to address the managerial and implementation issues involved in an improvement project. From a modeling point of view, this framework allows standardizing the formalisms of a large body of work on complex systems control or improvement stemming from many communities such as goal-driven engineering, industrial engineering and adaptive systems. From a pragmatic perspective, the challenge herein consists of developing a consensual transition from motivation to action, between managerial decisions and operational capabilities. As mentioned previously, the notion beyond this framework is that in any improvement project, executive managers are seeking the best for their companies, while operatives are doing their best. This challenge has constituted the source of our proposal.

Concerning the experiments, several directions may be envisaged to further enhance the algorithm’s performances. First, we aim at introducing new heuristics to improve the results and reduce the number of unsolved problems, especially for FRF (e.g. Pareto solutions obtained from DRF could be used to initialise the FRF resolution). Then, we intend to work on the optimisation of the implementation. The ongoing complexity of the algorithm is linear in space $O(|A| \times |N|)$ but quadratic in time for each call to CHOOSE or REDUCE $O(|A|^2 \times |N|)$. This means that its time complexity is about $O(|A|^2 \times |N|) \times ntc$. Our studies on the algorithm show that it could be reduced in time to $O(|A| \times |N|) \times ntc$ by preprocessing and sorting information of the digraph (Actions, Arcs and Performances). This one shot preprocessing could be done in $O(|A|^2 \times |N|)$ in time and space. This should significantly enhance the resolution time: instead of about 10 minutes to perform $ntc = 10^6$, 5 seconds would be sufficient on the same CPU.

References


\[^{3}\text{when } |A| > |N|.\]
\[^{4}\text{But will complexify the implementation a lot.}\]


