An Algebra of Pareto Points*

Marc Geilen, Twan Basten, Bart Theelen and Ralph Otten
Eindhoven University of Technology
Department of Electrical Engineering, Electronic Systems Group
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{m.c.w.geilen,a.a.basten,b.d.theelen,r.h.j.m.otten}@tue.nl

Abstract

Multi-criteria optimisation problems occur naturally in many engineering practices. Pareto analysis has proven to be a powerful tool to characterise potentially interesting realisations of a particular engineering problem. It is therefore used frequently for design-space exploration problems. Depending on the optimisation goals, one of the Pareto-optimal alternatives will be the optimal realisation. It often happens however, that partial design decisions have to be taken, leaving other aspects of the optimisation problem to be decided at a later stage, and that Pareto-optimal configurations have to be composed (dynamically) from Pareto-optimal configurations of components. These aspects are not supported by current analysis methods. This paper introduces a novel, algebraic approach to Pareto analysis. The approach is particularly designed to allow for describing incremental design decisions and composing sets of Pareto-optimal configurations. The algebra can be used to study the operations on Pareto sets and the efficient computation of Pareto sets and their compositions. The algebra is illustrated with a case-study based on transmitting an MPEG-4 video stream from a server to a hand-held device.

1 Introduction

In multi-criteria optimisation or decision problems there are multiple, potentially dependent and conflicting objectives. Depending on the relative importance of the objectives, different solutions may be preferred. The notion of Pareto optimality, named after Vilfredo Pareto who introduced it in [13] as '*maximum ophelimity*' in an economic system, states that a solution is optimal if it is impossible to find a solution which improves on one or more of the objectives without worsening any of them. If one solution is better in one objective than another solution and not worse in any other objectives, the latter is *dominated* by the former, which should always be preferred. Exploring the potential solutions to an optimisation problem without knowing a priori the exact optimisation criteria, then amounts to finding all solutions that are not dominated by any other (feasible) solution. This set of solutions is called the Pareto frontier and is guaranteed to contain all optimal solutions, whatever way the individual objectives are weighted relative to each other. To put it in other words, the Pareto frontier exactly captures the available trade-offs between the different objectives.

Multi-criteria optimisation problems appear frequently in a wide range of fields including business, economics and all kinds of engineering activities. In this paper, we briefly discuss some engineering applications, and in particular dynamic Quality-of-Service (QoS) management in wireless networks of embedded resource-constrained devices. Selecting an optimal configuration for a dynamic constellation of devices and available resources requires optimisation of power consumption, quality, timeliness, and so on. All of this needs to be done at run-time. The available configurations of the components themselves may be obtained from off-line Pareto analysis, of different architecture mappings for instance, for example by means of profiling. Some configurations may be good for efficient use of energy, others for obtaining a

*This work is supported by the IST - 004042 project, Betsy. This paper is a revised and extended version of [7]. It elaborates the theory in more detail providing among others proofs of all results and equational laws of compositions of Pareto sets. Additionally, the detailed MPEG-4 case-study of Section 8 is new.

higher quality, and so forth. The selection of a configuration at run-time requires composition of Pareto-optimal configurations of parts into Pareto-optimal configurations of a system. Some of these operations will have to be performed at run-time on resource-constrained devices. The number of configurations will have to be kept small and partial selection among Pareto optimal configurations may therefore be needed.

This paper introduces an algebraic approach to Pareto-optimal sets of configurations and defines operations on such sets. The algebra of sets of Pareto points provides a formal framework that allows to study compositions of Pareto sets and their properties, to define how to compute them and to apply cost functions to select optimal configurations or reduce the number of configurations to be stored. A new aspect is the use of partially ordered quantities as the parameters of solutions, which turns out to be the crucial point to obtain the compositionality needed for an algebra.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 gives motivation for the algebra introduced in this paper and introduces an example inspired by MPEG-4 video coding and decoding in a dynamic network context that is used throughout the paper to exemplify the concepts that are introduced. Section 4 formalises the well-known notions of Pareto dominance and Pareto optimality. In Section 5, we take a closer look at the cost functions that provide a ranking of Pareto points, their relationship with Pareto analysis and their use in intermediate, partial selection to reduce the size of Pareto sets. Section 6 then introduces the operators of the algebra and studies their properties, in particular with respect to Pareto optimality of the operands and of the composition results. For practical use of the algebra to compute sets of configurations, we look at some common combinations of the operators in Section 7. Section 8 elaborates the MPEG-4 example. Section 9 discusses computational and complexity issues and Section 10 concludes.

2 Related Work

Pareto analysis is used in many engineering practices [19]. In our own field of research for instance, we can mention its use in design-space exploration [10, 16], application mapping on embedded multiprocessor systems [15, 18] or architecture exploration for Systems-on-Chip (SoCs) [8]. Other applications include multidimensional optimisation problems in general, economics, mechanical engineering, electronic system design, power aware scheduling and SoC design. There is also a vast body of work on Pareto analysis and multi-objective optimisation and decision making in general (see e.g. [5] for an overview). This kind of work traditionally focusses on characterisation of the Pareto frontier based on the nature of the cost (objective) functions (typically continuous functions on real numbers) and algorithms for computing the Pareto frontier or approximations of the frontier [12, 21]. We do not look at particular objectives or cost functions, but rather look at the algebraic structure and properties, requiring only that an objective or cost function be monotone (which we show in this paper to be necessarily so for the Pareto approach to be adequate). The particular novelty of our approach lies in the fact that we pursue an incremental, compositional approach to determining Pareto sets, based on a rigourous algebra of well-defined operations on Pareto sets.

Algorithms to compute the Pareto frontier include among many other (see [4]) genetic or evolutionary algorithms [4, 20, 21], and tabu search algorithms [1, 9]. Algorithms to identify Pareto points in sets of configurations (also called Pareto minimisation) are introduced, although according to [19], literature on identifying Pareto sets is sparse. The algorithm of Bentley [2, 11] is asymptotically very efficient. [19] introduces a hybrid algorithm which combines it with a straightforward algorithm which is efficient for small sets. Data structures for storing sets of configurations have also been studied [14, 19]. In particular quad-trees have been introduced to manipulate such sets more efficiently, especially when the sets are large and the number of objectives is large. The behaviour of these algorithms is studied in [14, 19]. [19] also studies representative statistical distributions of random sets of configurations for studying such algorithms. The results of these efforts are also useful for implementing the operations of the algebra in this paper and reduction of the sets of configurations by approximation when their size becomes too large.

An algebraic characterisation of the Pareto dominance relation is presented in [17], which shows that the Pareto dominance relation is the only one satisfying certain axioms expected from such a relation. This is to the best of our knowledge, the only work that takes an algebraic approach to Pareto points. It does not introduce an algebra however, as we do in this paper.

3 Motivating Examples

To further motivate our algebra of Pareto points, we elaborate a bit on two potential applications in electronic system design. The two examples are of a complementary nature, each with their own specific requirements. We do expect however that the results are also useful to other engineering and maybe even non-engineering disciplines outside of electronic system design. The final subsection of this section introduces the running example that is used in an abstract form throughout the paper and that is detailed in Section 8.

3.1 Design Closure in SoC Design

Design closure for electronic design automation (EDA) tools means that users can specify a given function to be implemented on a chip, feed that to an EDA tool, and get, without further interaction, a design that meets all requirements concerning functionality, speed, size, power, yield, and other "costs" simultaneously. The EDA industry has thus-far concentrated on tools and techniques to achieve closure with respect to individual target design parameters at the lower levels of abstraction. Today's chip synthesis requires tools and methodologies for manipulating designs at higher levels of abstraction to meet a large variety of performance constraints. The algebra in this paper can support trade-offs between performance characteristics over many levels of abstraction.

Around 1980, designers realised that the complexity of chips would force them to use more than just a routing and occasionally a placement algorithm to find a starting point. It was obvious that the two tasks were heavily dependent on each other. Routing without a placement was inconceivable while at the same time a placement might render any routing infeasible. A solution to this dependency problem, named *wiring closure*, was found with a generalisation of placement called floor-plan design, slicing and floor-plan optimisation, resulting in a trade-off between the chip dimensions without design iterations. Later, a similar thing happened with the *timing closure* problem in which a trade-off between size and speed of gates is obtained. After timing, the primary concern for today's *SoC* devices is power consumption. The trade-off is between speed and energy efficiency of circuits.

In current practice, closure is achieved by Pareto style trade-offs. Every realisation has a certain combination of performance characteristics. The best Pareto point for a specific cost function is selected. Premature commitment to a particular realisation while unaware of consequences later in the design trajectory, implies the risk that the realisation turns out to be not the optimal one. Nowadays, there are many relevant characteristics, interacting with different levels of the design process. Choosing the 'best' one at a certain level may preclude optimal ones at later stages. Multidimensional Pareto analysis can help to solve these problems by its ability to partially resolve design decisions at a certain stage if it can be done safely, and use the results, compositionally, at later stages in the design. This way realisation options are not unnecessarily lost.

This example illustrates a typical off-line, design-time analysis scenario. Such analyses may involve large numbers of Pareto points and being off-line, the time needed for minimisation is not so important. Nevertheless, the most important challenge is to handle the large sets of Pareto points efficiently across multiple levels of abstraction. In contrast, for run-time analysis, the time for minimisation is crucial and for that, the number of Pareto points that need to be manipulated has to be limited. The on-line analysis is typically limited to one or a few levels of abstraction and may build upon partial optimisation results determined off-line. Such an example is presented in the next section.

3.2 Run-time Reconfiguration and QoS

To optimise Quality-of-Service and Quality-of-Experience, modern multimedia and consumer electronics systems need to adapt to dynamically changing circumstances and needs or desires of users. An ad-hoc formation of components needs to be configured dynamically by a QoS management system. To allow QoS management to use a component to its full extent, the component needs to expose its configuration options and trade-offs to the manager. The optimal working configuration is partly determined off-line by analysing possible realisations, and partly on-line by combining the trade-offs of all components to select a globally optimal configuration, given decision parameters such as user preferences or resource shortage.

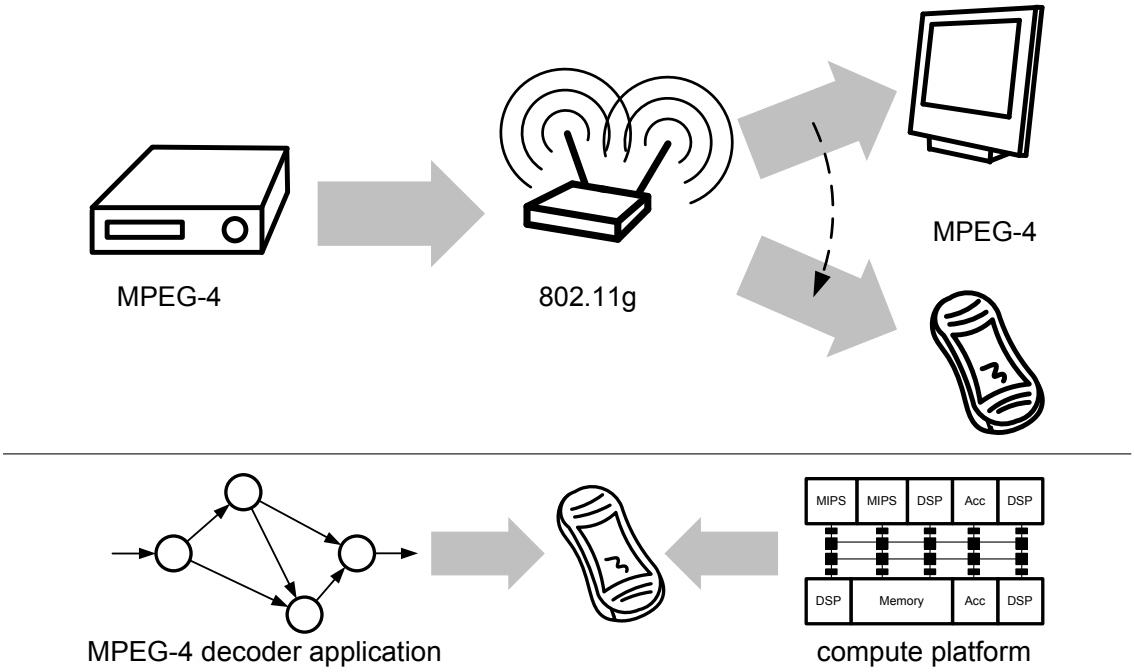


Figure 1: Wireless follow-me scenario

A component may expose its configuration options to the QoS manager in the form of a Pareto set. In this setting, doing an off-line design-space exploration for such components amounts to making the design decisions that lead to the best component, i.e. the realisation which has the best run-time trade-offs.

When Pareto sets are used at run-time to select working configurations, then the representation of the trade-offs must be very efficient and the run-time computation of the system trade-offs from component trade-offs and the run-time analysis of the optimal configuration need to be very fast.

3.3 A Wireless Follow-me Application

Figure 1 shows a simple application scenario for the approach developed in this paper. A server connected to a wired infrastructure encodes a video stream for transmission over a wireless network to a large television screen. We consider the moment that the user moves away from the screen and the system initiates a reconfiguration that moves the stream from the large screen to a hand-held device that the user can take with

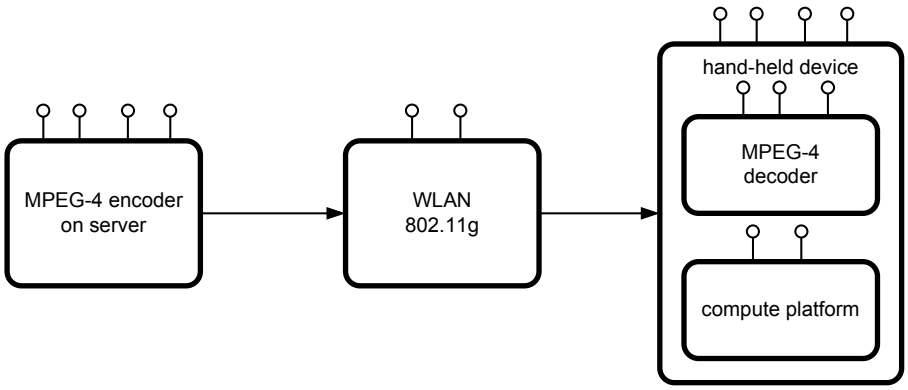


Figure 2: A wireless multimedia stream

him/her. The display devices decode the stream and display it on the screen. The hand-held itself consists of a hardware platform with different kinds of processing elements. The decoder and display applications are mapped onto this hardware platform and different mappings result in different characteristics (see also Figure 1). The resulting end-to-end delivery chain is depicted in Figure 2. Each of the components has its own parameters that can be set (illustrated by the small circles attached to the components) and a model of its corresponding trade-offs. An end-to-end quality manager can manipulate each of these parameters to satisfy overall, end-to-end objectives. For this, a model of the end-to-end trade-offs needs to be composed from the trade-off models of the individual components.

The following QoS-related trade-offs and properties play a role for this particular system.

- The server has a possibility to choose different encodings of the stream. The quality and required bandwidth of the generated stream depend on this. Energy consumption and the imposed computational load on the server are not considered an issue, as the server is assumed to have plenty of resources. In our example, we assume that the server runs an MPEG-4 encoder. Typical encoding parameters are the frame rate, frame size, quantisation parameter, GOP structure (Group-of-Pictures; types of encodings used for encoding individual frames and their order).

- The wireless network imposes a constraint on the available bandwidth; the bandwidth may fluctuate depending on environmental conditions. The wireless network can be used in different ways, for instance continuously, or in bursts. The latter could lead to energy savings in the hand-held, as the wireless network radio transceiver may be turned off in between these bursts to save energy.

- An MPEG-4 decoder running on the hand-held has a trade-off between quality and the bandwidth of the stream. This is essentially the same trade-off as in the server, but the decision needs to be taken together, since it obviously needs to be consistent. A consequence of the decoder parameter settings is having different computational efforts required from the platform for the decoding. This could be measured for instance in required processor cycles and memory usage.

- The platform of the hand-held is the hardware that should be able to deliver the computational effort demanded by the encoder. We assume that the platform provides different levels of computational power at different costs in terms of energy consumption, by exploiting possibilities for voltage and frequency scaling.

The properties of the decoder-to-platform mapping in the hand-held can be analysed off-line and the resulting options can be encoded into operating modes of the device. At run-time, when a user uses the device for displaying a video stream in a particular setting, a QoS management system can select the most appropriate mode of the hand-held and all other system components. When conditions change, e.g. drop in bandwidth or low battery, the QoS manager may initiate a reconfiguration procedure to adapt the working mode to the new circumstances.

We use the MPEG-4 example throughout this paper to illustrate the concepts introduced, and their practical application. Section 8 works out the example in detail.

4 Pareto Points

Pareto points describe the different qualities of possible system configurations, such as the bandwidth used, time needed for completion, energy consumption and so forth. We call these different dimensions of a configuration *quantities*. Quantities can be ordered in terms of better and worse, although we allow such an ordering to be partial. The latter allows us to consider, for example, the composition of execution time and energy consumption as a single, new quantity. Note that allowing partial orders means that the quantities can range from totally unordered (indifferent) to totally ordered (any two values can be compared), including situations in between.

Definition 1 (QUANTITY) *A quantity is a set Q with a partial order $\preceq_Q$. If the quantity is clear from the context we denote the order just by $\preceq$. If $\preceq_Q$ is total, the quantity is called* basic. *We assume that smaller values are preferred over larger ones. The irreflexive variant of $\preceq_Q$ is denoted $\prec_Q$.*

The quality level of the video encoding in the server of our example is, for instance, characterised with the quantity

$$SQuality = \{sq\_low, sq\_med, sq\_high\} \text{ with } sq\_high \preceq sq\_med \preceq sq\_low.$$

The quality settings are only ordered relative to each other. Being the result of human perception, it is inherently hard to quantify them with numbers. The corresponding server bit rate requirements are captured with the following quantity.

$$SBitRate = \{sb\_high, sb\_med, sb\_low\} \text{ with } sb\_low \preceq sb\_med \preceq sb\_high.$$

These values can be quantified, e.g., by profiling of the streams. For readability however, we do not use concrete numbers here. For the network bandwidth, we use the quantity

$$NBandwidth = \{nw\_low, nw\_med, nw\_high\} \text{ with } nw\_high \preceq nw\_med \preceq nw\_low.$$

Bit rates and network bandwidths are such that network bandwidth nw_high is required to support server bit rate sb_high, nw_med for sb_med and nw_low for sb_low. Note the difference between network bandwidth and server bit rate, where a *high* bandwidth, but *low* bit rate are preferred. From the network point of view, the higher the available bandwidth the better. The video encoding modes (representing a particular set of encoder settings) are captured in the unordered (i.e. also partially ordered) quantity:

$$EncodingMode = \{A, B, C\} \text{ with } \preceq = \{(A, A), (B, B), (C, C)\}.$$

Encoding modes cannot freely be determined by the server itself. Preferences among modes may originate from derived properties such as quality or available bandwidth.

System configurations are selected from a *configuration space* built up from quantities.

Definition 2 (CONFIGURATION SPACE) *A configuration space $\mathcal{S}$ is the Cartesian product $Q_1 \times Q_2 \times \ldots \times Q_n$ of a finite number of quantities.*

Definition 3 (CONFIGURATION) *A configuration $\bar{c} = (c_1, c_2, \ldots, c_n)$ is an element of configuration space $Q_1 \times Q_2 \times \ldots \times Q_n$. We use $\bar{c}(Q_k)$ or $\bar{c}(k)$ to denote c_k.*

Sets $\mathcal{C} \subseteq \mathcal{S}$ of configurations are used to represent the different options for realising a particular system or component. The configuration space of the server in the example could for instance be $EncodingMode \times SQuality \times SBitRate$. The feasible configurations could e.g. be

$$\{(A, sq\_low, sb\_low), (A, sq\_low, sb\_med), (B, sq\_med, sb\_med), (C, sq\_high, sb\_high)\}$$

which could be determined by profiling different realisations of the application.

A partial ordering of configurations is induced from the order on the individual quantities by a pointwise ordering.

Definition 4 (DOMINANCE) *If $\bar{c}_1, \bar{c}_2 \in \mathcal{S}$, then $\bar{c}_1 \preceq \bar{c}_2$ iff for every quantity Q_k of $\mathcal{S}$, $\bar{c}_1(Q_k) \preceq_{Q_k} \bar{c}_2(Q_k)$. If $\bar{c}_1 \preceq \bar{c}_2$, then $\bar{c}_1$ is said to* dominate *$\bar{c}_2$.*

Dominance of one configuration over another thus expresses the fact that the configuration is at least as good, because it is at least as good in each of the individual aspects. In this paper, for convenience, dominance is reflexive, i.e., a configuration dominates itself. The irreflexive variant is denoted as *strict dominance*, using the symbol $\prec$. For example, $(A, sq\_low, sb\_low) \prec (A, sq\_low, sb\_med)$ because the former provides the same quality at a lower bit rate for the same mode. In literature (e.g., [4]), the dominance relation is sometimes called weak dominance and strict dominance is used when a configuration is better in *all* quantities. Note that a configuration space with the dominance ordering is itself a quantity again, since it is a set with a partial order.

Often, we would like to remove from a set of configurations, elements that do not contribute interesting realisation options. A configuration that is strictly dominated by another one is typically not an interesting one. Sets of configurations that cannot be reduced without sacrificing potentially interesting realisations are called Pareto minimal (sometimes also Pareto optimal or Pareto efficient, or simply Pareto set).

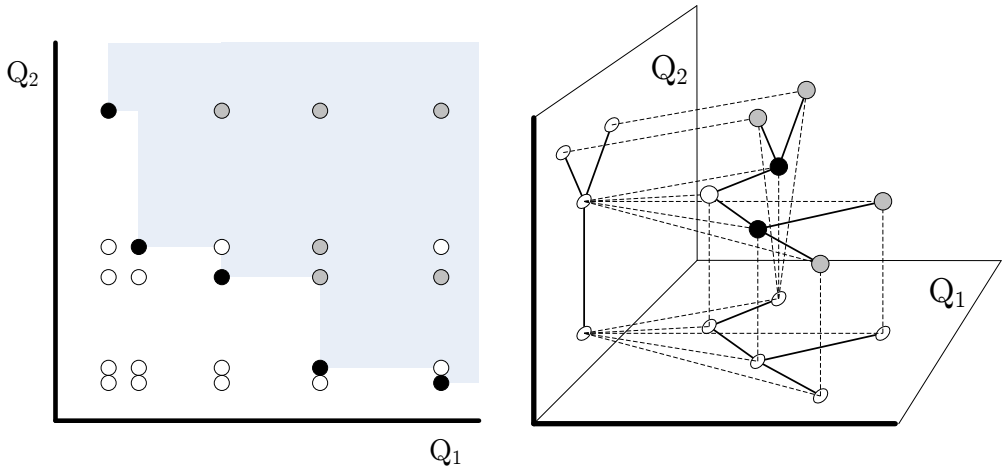


Figure 3: Compositionality using Pareto sets with partially ordered quantities

Definition 5 (PARETO MINIMAL) *A set $\mathcal{C}$ of configurations is said to be* Pareto minimal *iff for any $\bar{c}_1, \bar{c}_2 \in \mathcal{C}$, $\bar{c}_1 \not\prec \bar{c}_2$ (i.e., it is an anti-chain of the dominance relation).*

Minimality states that a set of configurations does not contain any strictly dominated configurations. In general, configurations in a set of configurations that are not strictly dominated by any other configuration, are called *Pareto points*. Thus, all individual elements of a Pareto minimal set of configurations are Pareto points. If we remove the dominated configuration from the configurations of the server, we obtain a Pareto minimal set:

$$\{(A, sq\_low, sb\_low), (B, sq\_med, sb\_med), (C, sq\_high, sb\_high)\}.$$

To compute Pareto minimal sets, we need to modify sets of configurations. The following definitions formalise that no interesting solutions are lost in such a process. First, the dominance of configurations is lifted to configuration sets.

Definition 6 (SET DOMINANCE) *A set $\mathcal{C}_1$ of configurations from configuration space $\mathcal{S}$ dominates a set $\mathcal{C}_2$ of configurations of $\mathcal{S}$, denoted as $\mathcal{C}_1 \preceq \mathcal{C}_2$, iff for every $\bar{c}_2 \in \mathcal{C}_2$ there is some $\bar{c}_1 \in \mathcal{C}_1$ such that $\bar{c}_1 \preceq \bar{c}_2$.*

We say that two sets of configurations are equally good if they dominate each other.

Definition 7 (PARETO EQUIVALENCE) *Two configuration sets $\mathcal{C}_1$ and $\mathcal{C}_2$ from configuration space $\mathcal{S}$ are Pareto equivalent, denoted $\mathcal{C}_1 \equiv \mathcal{C}_2$, iff they dominate each other: $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \preceq \mathcal{C}_1$.*

Pareto equivalence essentially claims that neither of the two sets contains a configuration that cannot be matched or improved upon by the other. It is easy to prove that $\equiv$ is indeed an equivalence relation. This notion of equivalence is often left implicit in the discussion of Pareto analysis. For our algebra however it is useful to make it precise. A Pareto minimal set of configurations can now also be characterised as follows. It is a set from which no configuration can be removed with an equivalent set as a result. Note that $\mathcal{C}_1$ and $\mathcal{C}_2$ need not be the same to be equivalent. However, equivalent sets can be reduced to the same Pareto minimal set while maintaining equivalence. In Section 6, we return to this point.

Example 1 *Figure 3(a) shows a configuration space of the basic quantities Q_1 and Q_2. The dots represent the configurations of the configuration space. The gray configurations are all dominated by at least one of the black configurations. If one has computed for instance the black set of configurations of a system, then the white configurations typically represent configurations that are infeasible (or not known to be feasible). The set of all black configurations is Pareto minimal and Pareto equivalent to the set of all black and gray configurations.*

Figure 3(a) shows a classical configuration space and Pareto points. A novel aspect of our approach is that it generalises quantities allowing them to be partially ordered. We tried to illustrate this case in Figure 3(b). To not clutter the picture, only a few configurations are shown. Again, the gray configurations are dominated by black configurations. For each gray configuraton there is a black one that is identical in one quantity, and better in the other. The two black configurations are incomparable, because they are incomparable in quantity Q_1.

Two equivalent *Pareto minimal sets* are necessarily the same.

Proposition 1 *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are two Pareto minimal sets of configurations and $\mathcal{C}_1 \equiv \mathcal{C}_2$, then $\mathcal{C}_1 = \mathcal{C}_2$.*

PROOF Let $\bar{c}_1 \in \mathcal{C}_1$. By dominance of $\mathcal{C}_2$ over $\mathcal{C}_1$, there is some $\bar{c}_2 \in \mathcal{C}_2$ such that $\bar{c}_2 \preceq \bar{c}_1$. Conversely, by dominance of $\mathcal{C}_1$ over $\mathcal{C}_2$, there is some $\bar{c}_1' \in \mathcal{C}_1$ such that $\bar{c}_1' \preceq \bar{c}_2 \preceq \bar{c}_1$. Since $\mathcal{C}_1$ is minimal, $\bar{c}_1' \preceq \bar{c}_1$ and $\bar{c}_1 \in \mathcal{C}_1$ implies that $\bar{c}_1' = \bar{c}_2 = \bar{c}_1$. Thus, we conclude that $\bar{c}_1 \in \mathcal{C}_2$ and hence that $\mathcal{C}_1 \subseteq \mathcal{C}_2$. Repeating the proof, switching the roles of $\mathcal{C}_1$ and $\mathcal{C}_2$, it follows that also $\mathcal{C}_2 \subseteq \mathcal{C}_1$ and hence, $\mathcal{C}_1 = \mathcal{C}_2$. □

The following theorem shows that all well-ordered sets of configurations have a unique minimal equivalent. This minimal equivalent set is often called its *Pareto frontier* and the elements are the Pareto points of the original set of configurations. A partial order $(D, \sqsubseteq)$ is called well-ordered if every chain (i.e. totally-ordered subset) contains a smallest element.

Theorem 1 *If $\mathcal{C}$ is a set of configurations and $(\mathcal{C}, \preceq)$ is well-ordered, then there is a unique Pareto minimal set $\mathcal{D}$ such that $\mathcal{D} \equiv \mathcal{C}$.*

PROOF We need to prove that (i) there exists an minimal set equivalent to $\mathcal{C}$ and (ii) it is unique. (i) For every configuration $\bar{c} \in \mathcal{C}$, we can pick a configuration $\bar{d}_{\bar{c}} \in \mathcal{C}$ that dominates $\bar{c}$, but is itself not strictly dominated by any configuration of $\mathcal{C}$, because $(\mathcal{C}, \preceq)$ is well-ordered. Let $\mathcal{D} = \{\bar{d}_{\bar{c}} \mid \bar{c} \in \mathcal{C}\}$. Then $\mathcal{D}$ is Pareto minimal and $\mathcal{D} \equiv \mathcal{C}$. (ii) Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be Pareto minimal sets of configurations and Pareto equivalent to $\mathcal{C}$. Then, since $\mathcal{D}_1 \equiv \mathcal{D}_2$, Proposition 1 shows that $\mathcal{D}_1 = \mathcal{D}_2$. □

For instance, the infinite set $\mathbb{N}$ of natural numbers has a Pareto minimal equivalent: $\{1\}$ and moreover, every set of natural numbers has a minimum (which is known as the Well Ordering Principle). The interval $(1, 2]$ of real numbers however, is not well-ordered. Such sets have to be avoided. As a direct consequence of the previous theorem and the fact that every finite partial order is well-ordered, we know that every finite set of configurations has a unique minimal equivalent.

Corollary 1 *For every finite set $\mathcal{C}$ of configurations, there is a unique Pareto minimal set $\mathcal{D}$ such that $\mathcal{D} \equiv \mathcal{C}$.*

In the remainder of this paper, we assume that sets of configurations are well-ordered.

As a final remark, note that the Cartesian product construct used in the definition of a configuration space induces an ordering of quantities in configuration spaces and sets. Usually, this ordering is artificial, in the sense that configuration spaces and sets that only differ in the ordering of quantities can be considered equal. This can be captured in the notion of permutation equivalence.

Definition 8 (PERMUTATION EQUIVALENCE) *Let $\mathcal{C}$ be a configuration set from configuration space $\mathcal{S}$, and let $\mathcal{C}_\pi$ and $\mathcal{S}_\pi$ be the configuration set and space obtained from $\mathcal{C}$ and $\mathcal{S}$ by reordering all the quantities according to permutation π. Set $\mathcal{C}$ and space $\mathcal{S}$ are said to be permutation equivalent to $\mathcal{C}_\pi$ and $\mathcal{S}_\pi$, denoted $\mathcal{C} =_p \mathcal{C}_\pi$ and $\mathcal{S} =_p \mathcal{S}_\pi$, respectively.*

5 Cost Functions

Pareto points represent potentially interesting system realisation alternatives. When the time comes to actually select one, single configuration to be realised, additional information is required to determine which configuration is best. This additional information is typically given as a cost function that can be applied to all the configurations and one with minimal cost is chosen.

5.1 Pareto Minimal Sets as Abstractions of Cost Functions

One way to look at Pareto points is as an abstraction of specific cost functions. With a clear objective, expressed as a cost function, an optimal configuration can be selected and alternative configurations are in principle irrelevant. One may choose to look at one single quantity, such as the energy consumption in the hand-held in our example, or the quality of the stream, but also at a weighted combination of both objectives. Sometimes however, one would like to resolve the decision only partially and let some independent objectives remain. In [15], for instance, a weighted sum of sequential and parallel execution times of an application is computed because the former is more relevant on a sequential platform, while the latter is more relevant on a parallel platform. The result is evaluated for different combinations of weights to obtain good solutions for both objectives. It would be more natural here to use a partially ordered cost function which retains parallel as well as sequential optimal execution times of solutions, without the need for such weight factors. Our framework supports such incremental, partial decision making using partially ordered quantities and cost functions with a partially ordered codomain.

An important observation is that the exact cost function is often unknown at a certain moment in the design. Pareto points represent exactly those configurations that can potentially be optimal under certain 'reasonable' cost functions. It does not make sense to take just any conceivable cost function into account, since then any configuration can potentially be optimal. A cost function is reasonable if it behaves monotonically with respect to the design's quantities. If a configuration dominates another configuration, then the cost should not get worse. Under this assumption, the Pareto points capture precisely the potentially interesting configurations. This is captured by the following two theorems proven below. Firstly, for a monotone cost function, a dominated configuration cannot have a better cost than the dominating configuration. (In other words, the Pareto points are sufficient.) Conversely, for any Pareto minimal set and a single configuration in this set, there exists a cost function for which that configuration has optimal cost. (Thus, all Pareto points are necessary.)

Definition 9 (MONOTONICITY) *A function $f : \mathcal{S} \to Q$ from configuration space $\mathcal{S}$ to quantity Q is* monotone *(or order-preserving) iff for any $\bar{c}_1, \bar{c}_2 \in \mathcal{S}$ such that $\bar{c}_1 \preceq \bar{c}_2$, $f(\bar{c}_1) \preceq_Q f(\bar{c}_2)$.*

Definition 10 (COST FUNCTION, OPTIMALITY) *A* cost function *on a configuration space $\mathcal{S}$ is a monotone function $f : \mathcal{S} \to Q$. A configuration $\bar{c}$ is called an* optimum in the set $\mathcal{C} \subseteq \mathcal{S}$ of configurations for cost *f iff for any configuration $\bar{c}' \in \mathcal{C}$, $f(\bar{c}) \preceq f(\bar{c}')$ (*the *optimum if it is unique) and $\bar{c}$ is called* optimal in $\mathcal{C}$ for *f iff for every configuration $\bar{c}' \in \mathcal{C}$, $f(\bar{c}') \not\prec f(\bar{c})$. Cost function f can be lifted to sets of configurations: $f(\mathcal{C}) = \{f(\bar{c}) \mid \bar{c} \in \mathcal{C}\}$.*

In contrast with traditional Pareto analysis approaches, we allow the cost function to return a value in a partially ordered domain. This means that application of the cost function adds information about the preference of Pareto points, but need not make a definitive selection of a single optimum configuration. Note that any configuration space with its dominance relation is a quantity. Hence, any monotone transformation on configuration spaces can be considered as a cost function.

Multiple configurations may be optimal and an optimum need not exist. This enables an incremental compositional way of computing trade-offs allowing for partial selection decisions to be taken at different places in the process. Note also that monotonicity of the cost function implies that the cost of two configurations with a dominance relationship can always be compared (the dominating configuration must have the same or better cost). The following two theorems (which are variations of the two Fundamental Theorems of Welfare Economics [13]) prove that the Pareto minimal sets of configurations capture precisely the required information to guarantee that an optimal configuration can be selected for any arbitrary monotone cost function.

Theorem 2 *(*SUFFICIENCY OF PARETO SETS*) Let $f : \mathcal{S} \to Q$ be a cost function. If $\bar{c}_1, \bar{c}_2 \in \mathcal{S}$ with $\bar{c}_1 \preceq \bar{c}_2$, then $f(\bar{c}_1) \preceq f(\bar{c}_2)$.*

PROOF Follows immediately from monotonicity of f. □

This theorem shows that the minimal Pareto set is sufficient to capture all relevant configurations under any cost function. The following theorem states conversely that all configurations of a Pareto minimal set are necessary.

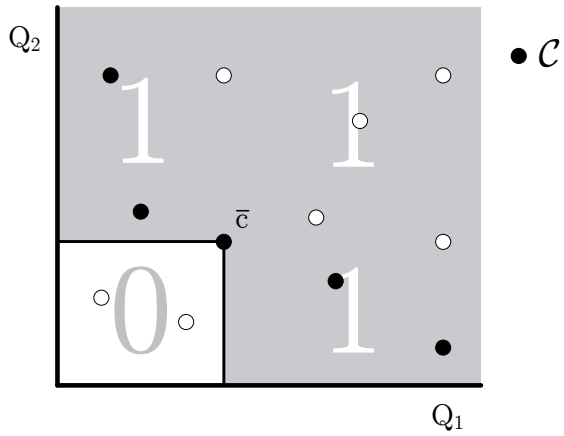


Figure 4: Cost function used in the proof of Theorem 3

Theorem 3 *(*NECESSITY OF PARETO SETS*) Let $\mathcal{C}$ be a Pareto minimal set of configurations of $\mathcal{S}$ and $\bar{c} \in \mathcal{C}$. Then, there exists a cost function f on $\mathcal{S}$ such that $\bar{c}$ is optimal (and even the optimum) for f.*

PROOF Let $f : \mathcal{S} \to \{0, 1\}$ (with $0 \preceq 1$) be as follows (illustrated with Figure 4). $f(\bar{c}') = 0$ for every $\bar{c}' \in \mathcal{S}$ such that $\bar{c}' \preceq \bar{c}$ (thus including $\bar{c}$ itself) and $f(\bar{c}') = 1$ for every other configuration $\bar{c}'$. Then (i) f is a cost function and (ii) $\bar{c}$ is optimal in $\mathcal{C}$. (i) We have to show that f is monotone. Assume the contrary. Then there are $\bar{c}_1, \bar{c}_2 \in \mathcal{C}$, such that $\bar{c}_1 \preceq \bar{c}_2$, $f(\bar{c}_1) = 1$ and $f(\bar{c}_2) = 0$. Then, $\bar{c}_1 \preceq \bar{c}_2$ and $\bar{c}_2 \preceq \bar{c}$ (because $f(\bar{c}_2) = 0$) and by transitivity, $\bar{c}_1 \preceq \bar{c}$, contradicting the fact that $f(\bar{c}_1) = 1$ and thus that f is not monotone. (ii) Since $\mathcal{C}$ is Pareto minimal, there is no $\bar{c}' \in \mathcal{C}$ such that $\bar{c}' \prec \bar{c}$ and hence, $f(\bar{c}') = 1$ for all other $\bar{c}' \in \mathcal{C}$. Thus, $\bar{c}$ is the optimum in $\mathcal{C}$ and hence optimal. □

5.2 Priority

An example of a practically useful cost function that may only partially resolve the choice of an optimal configuration is prioritisation. The size of a set of configurations can be reduced by introducing a prioritisation on (some of) the quantities. If a configuration is strictly better in quantity A, and A is given priority over quantity B, then that configuration will always be preferred, no matter what quantity B is of the other configuration. Only if the configurations are the same or incomparable on quantity A, quantity B is taken into account. For instance, a shared multimedia infrastructure has to select an optimal configuration in case there are multiple users with different preferences. The users may establish amongst themselves a priority relation indicating the relative importance of their personal preferences (quality as experienced by one user has priority over the quality experienced by the other user). This information can be used by the run-time manager to select a concrete working configuration. The priority order does not have to be total, but can also be partial ('user A has priority over user B, but between user A and C, we don't know').

Definition 11 (PRIORITY) *Let $\trianglelefteq$ be a well-ordered partial ordering on the quantities $\{Q_i \mid 1 \leq i \leq n\}$, indicating their relative priorities and let $\overline{Q}$ be the Cartesian product $Q_1 \times \ldots \times Q_n$. Define a cost function $prio : \overline{Q} \to R$ as follows; the quantity R is the same set $\overline{Q}$ and $prio$ is the identity function, but R has the following order: $\bar{r}_1 \preceq_R \bar{r}_2$ iff for every i, $1 \leq i \leq n$, $\bar{r}_1(Q_i) \preceq_{Q_i} \bar{r}_2(Q_i)$ or there is some j, such that $Q_j \trianglelefteq Q_i$ and $\bar{r}_1(Q_j) \prec \bar{r}_2(Q_j)$.*

Note that the $prio$ cost function can be seen as a transformation on the given configuration space, as explained above. It does not change the configurations or assign cost values to it, but it adds information about preferences among the Pareto points. When interpreting R as a quantity instead of a configuration space, the $prio$ cost function 'collapses' the structure of the original configuration space to the single new quantity R.

The following two propositions formally prove that Definition 11 defines indeed a cost function.

Proposition 2 *The relation $\preceq_R$ as defined in Definition 11 is a partial order.*

PROOF We have to prove (i) reflexivity, (ii) antisymmetry and (iii) transitivity of $\preceq_R$. Let the rank of quantity Q be defined as the length of the largest chain on the priorities with least upper bound Q. (i) Reflexivity follows trivially from the definition. (ii) Let $\bar{r}_1 \preceq \bar{r}_2$ and $\bar{r}_2 \preceq \bar{r}_1$. Antisymmetry can be proved by showing by induction on the rank of quantities, that for every quantity Q, $\bar{r}_1(Q) = \bar{r}_2(Q)$, because $\bar{r}_1$ and $\bar{r}_2$ are the same for all lower ranked quantities and hence must also be the same for Q. (iii) Transitivity. Let $\bar{r}_1 \preceq \bar{r}_2$ and $\bar{r}_2 \preceq \bar{r}_3$. We have to show that $\bar{r}_1 \preceq \bar{r}_3$, i.e. that for every quantity Q_i, $\bar{r}_1(Q_i) \preceq_{Q_i} \bar{r}_3(Q_i)$ or there is some j, $Q_j \trianglelefteq Q_i$, such that $\bar{r}_1(Q_j) \prec \bar{r}_3(Q_j)$. If the rank of Q_i is 1, then there is no quantity of smaller rank, i.e., with higher priority. Then, $\bar{r}_1(Q_i) \preceq \bar{r}_2(Q_i)$ and $\bar{r}_2(Q_i) \preceq \bar{r}_3(Q_i)$ and hence $\bar{r}_1(Q_i) \preceq \bar{r}_3(Q_i)$. Let the rank of $Q_i > 1$. Assume that $\bar{r}_1(Q_i) \not\preceq_{Q_i} \bar{r}_3(Q_i)$. Then $\bar{r}_1(Q_i) \not\preceq_{Q_i} \bar{r}_2(Q_i)$ or $\bar{r}_2(Q_i) \not\preceq_{Q_i} \bar{r}_3(Q_i)$. We have to show that there is some j, $Q_j \trianglelefteq Q_i$, such that $\bar{r}_1(Q_j) \prec \bar{r}_3(Q_j)$. Let Q_j be a smallest ranked quantity $Q_j \trianglelefteq Q_i$ and with $j \neq i$, such that $\bar{r}_1(Q_j) \prec \bar{r}_2(Q_j)$ or $\bar{r}_2(Q_j) \prec \bar{r}_3(Q_j)$. Such a quantity exists, because $\bar{r}_1(Q_i) \not\preceq_{Q_i} \bar{r}_2(Q_i)$ or $\bar{r}_2(Q_i) \not\preceq_{Q_i} \bar{r}_3(Q_i)$. Then, $\bar{r}_1(Q_j) \preceq_{Q_j} \bar{r}_2(Q_j)$ and $\bar{r}_2(Q_j) \preceq_{Q_j} \bar{r}_3(Q_j)$ (since the second clause of the definition of the order $\preceq_R$ cannot hold, otherwise, Q_j would not be of smallest rank, the first one must hold) and $\bar{r}_1(Q_j) \prec_{Q_j} \bar{r}_2(Q_j)$ or $\bar{r}_2(Q_j) \prec_{Q_j} \bar{r}_3(Q_j)$, from which it follows that $\bar{r}_1(Q_j) \prec_{Q_j} \bar{r}_3(Q_j)$. □

Proposition 3 *The function $prio$ is monotone.*

PROOF Let $\bar{c}_1$ and $\bar{c}_2$ be configurations of $Q_1 \times \ldots \times Q_n$ such that $\bar{c}_1 \preceq \bar{c}_2$. We know that $prio(\bar{c}_1) = \bar{c}_1$ and $prio(\bar{c}_2) = \bar{c}_2$ and we have to show that $\bar{c}_1 \preceq_R \bar{c}_2$, i.e., that for every i, $1 \leq i \leq n$, $\bar{c}_1(Q_i) \preceq_{Q_i} \bar{c}_2(Q_i)$ or there is some j, such that $Q_j \trianglelefteq Q_i$ and $\bar{c}_1(Q_j) \prec \bar{c}_2(Q_j)$. This is the case, since for every i, $1 \leq i \leq n$, $\bar{c}_1(Q_i) \preceq_{Q_i} \bar{c}_2(Q_i)$. □

The more priority relations are introduced, the fewer optimal potential solutions remain. When basic quantities are totally ordered in priority, then the configurations will be totally ordered and a unique optimum is obtained. Note that the monotonicity result formally shows that priorities only add ordering to the configurations and can never remove them; if $\bar{c}_1 \preceq \bar{c}_2$, then also $\bar{c}_1 \preceq_R \bar{c}_2$.

Proposition 4 *If $\trianglelefteq$ is a total order and the quantities of the configuration space are basic (totally ordered), then the resulting priority relation $\preceq_R$ is a total order.*

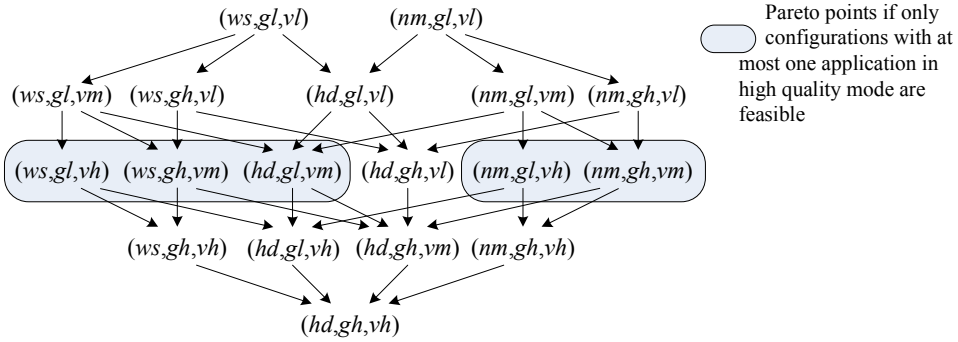
PROOF Assume towards a contradiction, that $\bar{r}_1 \not\preceq_R \bar{r}_2$ and $\bar{r}_2 \not\preceq_R \bar{r}_1$. Since $\bar{r}_1 \neq \bar{r}_2$, there is a smallest ranked quantity Q such that $\bar{r}_1(Q) \neq \bar{r}_2(Q)$, but then, because Q is basic, $\bar{r}_1(Q) < \bar{r}_2(Q)$ or $\bar{r}_2(Q) < \bar{r}_1(Q)$, and because Q has higher priority than any other Q' for which $\bar{r}_1$ and $\bar{r}_2$ are different, $\bar{r}_1 \preceq_R \bar{r}_2$ or $\bar{r}_2 \preceq_R \bar{r}_1$. □

The following example illustrates the concept of priorities, and hence cost functions. It illustrates the process of partial decision making and Proposition 4. In particular, it shows that both priorities and all quantities have to be totally ordered in order to guarantee a total order on all possible configurations.

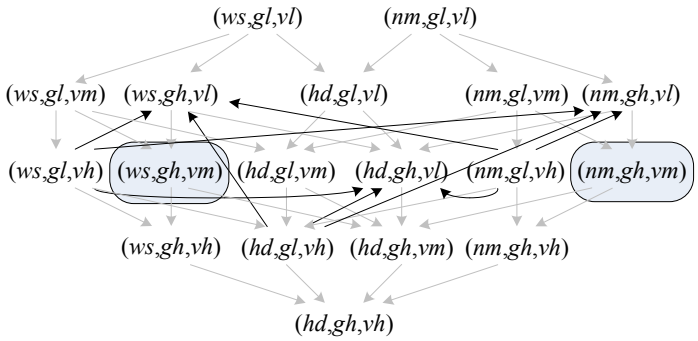
Example 2 *Consider a variant of our running example, in which the server provides not only video but also TV and gaming. It turns out that the server can only run two applications at the highest quality level if the third one is not running. However, if all three applications are running, only one can run in highest quality. Therefore, we are interested in the behaviour and the priorities if all three applications are running, and decide to create a model.*

Suppose that the TV supports three quality modes, high definition, normal, and wide screen, modelled through quantity $TV = \{hd, nm, ws\}$ with $hd \preceq nm$, $hd \preceq ws$, and no ordering between nm and ws. Video has three modes, high, medium, and low, modelled by quantity $Video = \{vh, vm, vl\}$ with $vh \preceq vm \preceq vl$. Gaming has two quality modes, high and low, modelled by quantity $Gaming = \{gh, gl\}$ with $gh \preceq gl$. Figure 5(a) shows all the potential (feasible and infeasible) configurations in the combined configuration space $TV \times Gaming \times Video$. It also shows the dominance relationships between the configurations in the form of arrows, where a sequence of arrows from configuration $\bar{a}$ to $\bar{b}$ indicates that $\bar{a}$ is dominated by $\bar{b}$; whenever dominance is implied by reflexivity and transitivity of other arrows that have already been drawn, the arrow is left out of the picture to improve readability. It also shows the Pareto points assuming that only configurations with at most one application in the highest quality mode is feasible.

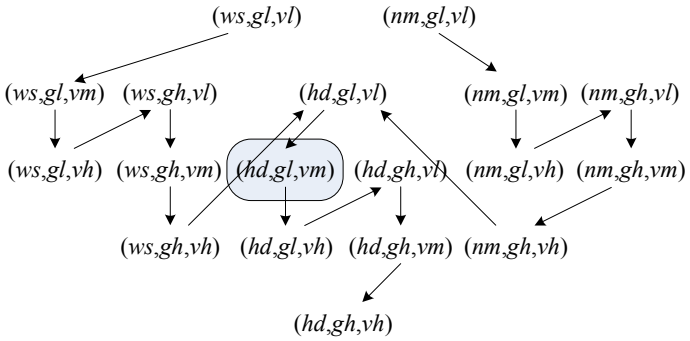
Since the youngest son of the family is a fanatic gamer, it is decided to give gaming the highest priority (while not specifying any priority among TV and video), i.e., $Gaming \trianglelefteq TV$ and $Gaming \trianglelefteq Video$. Figure 5(b) shows the effect on the configuration space. The black arrows visualise the extra dominance relationships added by this priority function.



(a) No priorities



(b) The *Gaming* quantity has priority over *TV* and *Video*



(c) *TV* has priority over *Gaming* and *Video*, and *Gaming* over *Video*

Figure 5: Priorities

However, the result is not satisfactory to dad, who prefers TV. Priority settings are changed: $TV \trianglelefteq Gaming \trianglelefteq Video$. *The configuration space becomes almost totally ordered. Only the configurations with normal and wide screen TV modes are still incomparable. Figure 5(c) shows the result.*

Sometimes dad allows his son to play games at the highest quality level, even if he is watching TV. He decides that he then prefers wide screen images over normal images, and adapts the settings accordingly, i.e., the ordering $ws \preceq nm$ *is added to the order of quantity* TV. *The configuration space becomes now totally ordered. Considering Figure 5(c), the only change is that the arrow from* (nm, gh, vh) *changes from* (hd, gl, vl) *to* (ws, gl, vl).

5.3 Derived Quantities

Cost functions as defined in Definition 10, with the cost codomain of functions generalised to partially ordered quantities, are a very versatile and powerful concept. Consider for example the following. Sometimes, the quantities of configurations refer to specific or detailed information. From that detailed information one may derive higher-level quantities or information. For instance, a configuration may contain the latencies of each of the individual components of a multimedia delivery chain. The end-to-end latency can be computed by adding them up and adding the result as a new quantity of the configuration. Similarly, total power consumption may be computed from parts, or a suitable notion of perceived quality can be estimated from measurable quantities such as signal-to-noise ratio.

We consider the following situation in general. Given a set $\mathcal{C}$ of configurations in a configuration space $\mathcal{S}$ and a function $f : \mathcal{S} \to Q$ for some quantity Q. We derive a new set of configurations $\mathcal{C}_f$ in configuration space $\mathcal{S} \times Q$ by adding one quantity derived from the existing quantities:

$$\mathcal{C}_f = \{(c_1, \ldots, c_n, f(c_1, \ldots, c_n)) \mid (c_1, \ldots, c_n) \in \mathcal{C}\}.$$

The obvious question is now: is this approach consistent with Pareto point analysis? We want to restrict the points in the original set of configurations to Pareto points and forget about dominated points. As a consequence, we would not like to have that if configuration $\bar{c}_1 \preceq \bar{c}_2$, then $f(\bar{c}_1) \not\preceq f(\bar{c}_2)$, because this could introduce new Pareto points from previously dominated points. When this would be allowed, the Pareto points are an inappropriate representation of the potentially optimal configurations. Hence, the answer to the above question is that this is only allowed using monotone functions, or in other words cost functions.

It is easy to see that summing individual latencies to end-to-end latency (or power consumptions to total power consumption) is monotone. Also multiplication and minimum or maximum operations are often useful and are monotone. If we want to used a derived property for estimating a measure of perceived quality from one or more objective quality measures, it only works if improved objective quality aspects can never worsen the perceived quality.

The examples of the priority cost function of the previous subsection and the addition of derived quantities to a configuration set explained in this subsection illustrate that a large class of cost functions can be seen as transformations on Pareto spaces, which fits elegantly in an algebraic framework. Note however that cost functions are in general not yet algebraic operators, because they usually cannot be applied to arbitrary configuration sets. It is a prerequisite for operators in an algebra that they can be applied to arbitrary elements from the domain of the algebra, configuration sets in our case. The concept of addition of derived quantities shows one other important point. To benefit from the Pareto approach, it is essential that dominated points can be removed from configuration sets at all times without negative consequences. This is also essential for the algebraic operators that we introduce in the next section.

6 Operations of the Algebra

In this section, we introduce operations on sets of configurations. With these operations we define an algebra of Pareto points that can be used to compute or reason about Pareto points of composite systems. We define the operations, show how they can be applied in practical problems and study their properties, in particular regarding Pareto minimality of operands and results.

6.1 Minimisation

The first operator takes a set of configurations and returns the unique equivalent Pareto minimal set (the Pareto frontier).

Definition 12 (MINIMISATION) *Let $\mathcal{C}$ be a set of configurations of configuration space $\mathcal{S}$ and $(\mathcal{C}, \preceq)$ well-ordered, then $\min(\mathcal{C})$ denotes the unique, Pareto equivalent and Pareto minimal set of configurations.*

Its use in practice stems from the fact that a minimal representation of a component's trade-offs makes it easier and more efficient to manipulate, at design-time or at run-time. From Proposition 1 and Theorem 1, the following corollaries follow immediately. (Recall that we assume sets of configurations to be well-ordered.)

Corollary 2 $\mathcal{C}_1 \equiv \mathcal{C}_2$ *iff* $\min(\mathcal{C}_1) = \min(\mathcal{C}_2)$.

Corollary 3 $\min(\mathcal{C}) \equiv \mathcal{C}$.

Corollary 4 $\min(\mathcal{C}) \subseteq \mathcal{C}$.

Finally, we observe that $\min$ is a projection, applying minimisation multiple times does not have any additional effect.

Corollary 5 $\min(\min(\mathcal{C})) = \min(\mathcal{C})$.

Minimisation is the key to keeping the size of a component's description or the number of configurations that we need to consider as small as possible. We want to answer two important questions about the operators we introduce in the remainder. The first asks whether an operator allows minimisation of its operands. Let f be an operator with n operands. f supports minimisation of its operands if for all configuration sets $\mathcal{C}_1, \ldots, \mathcal{C}_n$,

$$\mathrm{f}(\mathcal{C}_1, \ldots, \mathcal{C}_n) \equiv \mathrm{f}(\min(\mathcal{C}_1), \ldots, \min(\mathcal{C}_n)). \tag{1}$$

By minimising operands, which are often intermediate results in the composition process, before applying f, we will not loose optimal configurations. Every practical operator should satisfy property (1), for if this is not the case, then the Pareto minimal set is not an adequate abstraction of the component's configurations. To determine optimal configurations one would have to try *all* configurations which is often infeasible. If the answer to the first question is positive, the second question is whether an operator preserves minimality. This translates to the following property.

$$\min(\mathrm{f}(\mathcal{C}_1, \ldots, \mathcal{C}_n)) = \mathrm{f}(\min(\mathcal{C}_1), \ldots, \min(\mathcal{C}_n)). \tag{2}$$

If property (2) holds, then minimisation after applying operator f is unnecessary. The result is always minimal if the operands have been minimised, which saves time in computing the results of the operation.

In the remainder, instead of using property (1), we study the somewhat stronger property that Pareto equivalence is a congruence with respect to the operator. If $\mathcal{C}'_i \equiv \mathcal{C}_i$ for all $1 \leq i \leq n$, then

$$\mathrm{f}(\mathcal{C}'_1, \ldots, \mathcal{C}'_n) \equiv \mathrm{f}(\mathcal{C}_1, \ldots, \mathcal{C}_n). \tag{3}$$

To show this, it is sufficient to show that the operator preserves dominance. If $\mathcal{C}'_i \preceq \mathcal{C}_i$ for all $1 \leq i \leq n$, then

$$\mathrm{f}(\mathcal{C}'_1, \ldots, \mathcal{C}'_n) \preceq \mathrm{f}(\mathcal{C}_1, \ldots, \mathcal{C}_n). \tag{4}$$

Property (1) follows directly from (3) or (4), as $\min(\mathcal{C}) \equiv \mathcal{C}$ for any $\mathcal{C}$.

6.2 Free Product

A system often consists of multiple components that each have their own configurations and corresponding trade-offs. To combine these components to systems, we need to derive the configurations and trade-offs of the system from the components. The *free product* describes the combination and assumes that components can be configured entirely independently. Constraints on the composition can be introduced later with other operators.

Definition 13 (FREE PRODUCT) *Let $\mathcal{C}_1$ be a set of configurations of configuration space $\mathcal{S}_1$ and $\mathcal{C}_2$ a set of configurations of space $\mathcal{S}_2$. Then, the (free) product is the Cartesian product $\mathcal{C}_1 \times \mathcal{C}_2$ in the configuration space $\mathcal{S}_1 \times \mathcal{S}_2$. When $\bar{c}_1 \in \mathcal{C}_1$ and $\bar{c}_2 \in \mathcal{C}_2$, we use $\bar{c}_1 \cdot \bar{c}_2$ to denote the corresponding configuration in $\mathcal{C}_1 \times \mathcal{C}_2$.*

The configuration spaces of the server and the network of our example can be combined with the product operator. Assume that the network conditions are currently such that the set of available configurations is $\{nw\_low, nw\_med\}$. The (optimal) server configurations are $\{(A, sq\_low, sb\_low), (B, sq\_med, sb\_med), (C, sq\_high, sb\_high)\}$. The free product then yields:

$$\{(A, sq\_low, sb\_low, nw\_low), (B, sq\_med, sb\_med, nw\_low), (C, sq\_high, sb\_high, nw\_low),$$
$$(A, sq\_low, sb\_low, nw\_med), (B, sq\_med, sb\_med, nw\_med), (C, sq\_high, sb\_high, nw\_med)\}.$$

Some of these may not be feasible; the network bandwidth must be sufficient to carry the stream. This can be enforced with the constraint operator introduced in Section 6.4.

The product operator preserves dominance.

Proposition 5 (DOMINANCE PRESERVATION) *If $\mathcal{C}_1' \preceq \mathcal{C}_1$ and $\mathcal{C}_2' \preceq \mathcal{C}_2$, then $\mathcal{C}_1' \times \mathcal{C}_2' \preceq \mathcal{C}_1 \times \mathcal{C}_2$.*

PROOF Let $\bar{c} \in \mathcal{C}_1 \times \mathcal{C}_2$ and $\bar{c}_1 \in \mathcal{C}_1$ and $\bar{c}_2 \in \mathcal{C}_2$ such that $\bar{c} = \bar{c}_1 \cdot \bar{c}_2$. Because $\mathcal{C}_1' \preceq \mathcal{C}_1$ and $\mathcal{C}_2' \preceq \mathcal{C}_2$, there exist $\bar{c}_1' \in \mathcal{C}_1'$ and $\bar{c}_2' \in \mathcal{C}_2'$ such that $\bar{c}_1' \preceq \bar{c}_1$ and $\bar{c}_2' \preceq \bar{c}_2$. Thus, $\bar{c}_1' \cdot \bar{c}_2' \in \mathcal{C}_1' \times \mathcal{C}_2'$ and $\bar{c}_1' \cdot \bar{c}_2' \preceq \bar{c}$. □

As explained in Section 6.1, it follows immediately that the congruence and intermediate minimisation properties hold.

Corollary 6 (CONGRUENCE) *If $\mathcal{C}_1' \equiv \mathcal{C}_1$, $\mathcal{C}_2' \equiv \mathcal{C}_2$, then $\mathcal{C}_1' \times \mathcal{C}_2' \equiv \mathcal{C}_1 \times \mathcal{C}_2$.*

Corollary 7 (OPERAND MINIMISATION) $\mathcal{C}_1 \times \mathcal{C}_2 \equiv \min(\mathcal{C}_1) \times \min(\mathcal{C}_2)$.

Additionally, the product operator preserves minimality.

Proposition 6 (MINIMALITY PRESERVATION) $\min(\mathcal{C}_1 \times \mathcal{C}_2) = \min(\mathcal{C}_1) \times \min(\mathcal{C}_2)$.

PROOF ($\subseteq$) From Corollaries 3 and 6 it follows that $\min(\mathcal{C}_1 \times \mathcal{C}_2) \equiv \mathcal{C}_1 \times \mathcal{C}_2 \equiv \min(\mathcal{C}_1) \times \min(\mathcal{C}_2)$ and hence with Corollaries 2 and 4 that $\min(\mathcal{C}_1 \times \mathcal{C}_2) = \min(\min(\mathcal{C}_1) \times \min(\mathcal{C}_2)) \subseteq \min(\mathcal{C}_1) \times \min(\mathcal{C}_2)$. ($\supseteq$) Let $\bar{c} \in \min(\mathcal{C}_1) \times \min(\mathcal{C}_2)$. Then $\bar{c} = \bar{c}_1 \cdot \bar{c}_2$ for some $\bar{c}_1 \in \min(\mathcal{C}_1)$ and $\bar{c}_2 \in \min(\mathcal{C}_2)$. Assume that there exists some $\bar{c}' \in \mathcal{C}_1 \times \mathcal{C}_2$ such that $\bar{c}' \prec \bar{c}$ and $\bar{c}' = \bar{c}_1' \cdot \bar{c}_2'$ for some $\bar{c}_1' \in \mathcal{C}_1$ and $\bar{c}_2' \in \mathcal{C}_2$. Then $\bar{c}_1' \prec \bar{c}_1$ or $\bar{c}_2' \prec \bar{c}_2$ and hence $\bar{c}_1 \notin \min(\mathcal{C}_1)$ or $\bar{c}_2 \notin \min(\mathcal{C}_2)$. This contradiction shows that such a $\bar{c}'$ does not exist and therefore $\bar{c} \in \min(\mathcal{C}_1 \times \mathcal{C}_2)$. □

It is interesting to see whether the algebraic operations introduced in our algebra satisfy desirable properties, such as for example commutativity and associativity, and what equalities we can derive for expressions in the algebra. The following proposition summarises some interesting properties of the free product.

Proposition 7 *Let $\mathcal{C}, \mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ be arbitrary configuration sets.*

$\mathcal{C}_1 \times \mathcal{C}_2 =_p \mathcal{C}_2 \times \mathcal{C}_1$ (COMMUTATIVITY)

$(\mathcal{C}_1 \times \mathcal{C}_2) \times \mathcal{C}_3 = \mathcal{C}_1 \times (\mathcal{C}_2 \times \mathcal{C}_3)$ (ASSOCIATIVITY)

$\varnothing \times \mathcal{C} = \mathcal{C} \times \varnothing = \varnothing$ (ZERO ELEMENT)

PROOF Straightforward from set theory. □

Note that commutativity only holds with respect to permutation equivalence and not for set equality; the order of the quantities is different in both sides of the equation. The second property is the expected associativity property, while the third property states that the empty set of configurations $\varnothing$ is a zero element for the free product.

6.3 Alternatives

If configuration sets $\mathcal{C}_1$ and $\mathcal{C}_2$ from configuration space $\mathcal{S}$ describe alternative possible realisations, then the choice between both types of realisations is characterised as follows.

Definition 14 (ALTERNATIVES) *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ both be a set of configurations of configuration space $\mathcal{S}$. Then $\mathcal{C}_1 \cup \mathcal{C}_2$ is a set of configurations of $\mathcal{S}$ called the set of alternatives of $\mathcal{C}_1$ and $\mathcal{C}_2$.*

The hardware platform of the hand-held device may have a choice between two processing elements to realise the stream decoder. The different elements may offer different amounts of processing or storage capacity. If we have Pareto sets characterising the performance of the decoders on the individual processing elements, then their union characterises the possible ways in which the hand-held platform can execute the stream decoder.

Proposition 8 (DOMINANCE PRESERVATION) *If $\mathcal{C}'_1 \preceq \mathcal{C}_1$ and $\mathcal{C}'_2 \preceq \mathcal{C}_2$, then $\mathcal{C}'_1 \cup \mathcal{C}'_2 \preceq \mathcal{C}_1 \cup \mathcal{C}_2$.*

PROOF Let $\bar{c} \in \mathcal{C}_1 \cup \mathcal{C}_2$. If $\bar{c} \in \mathcal{C}_1$, then there is some $\bar{c}' \in \mathcal{C}'_1$ that dominates it, because $\mathcal{C}'_1 \preceq \mathcal{C}_1$ and hence, $\bar{c}' \in \mathcal{C}'_1 \cup \mathcal{C}'_2$. The case $\bar{c} \in \mathcal{C}'_2$ is symmetrical. □

We know that from the previous proposition, the congruence property follows immediately.

Corollary 8 (CONGRUENCE) *If $\mathcal{C}'_1 \equiv \mathcal{C}_1$, $\mathcal{C}'_2 \equiv \mathcal{C}_2$, then $\mathcal{C}'_1 \cup \mathcal{C}'_2 \equiv \mathcal{C}_1 \cup \mathcal{C}_2$.*

Corollary 9 (OPERAND MINIMISATION) $\mathcal{C}_1 \cup \mathcal{C}_2 \equiv \min(\mathcal{C}_1) \cup \min(\mathcal{C}_2)$.

The alternative does not preserve minimality. It is generally not true that $\min(\mathcal{C}_1 \cup \mathcal{C}_2) = \min(\mathcal{C}_1) \cup \min(\mathcal{C}_2)$. This is the case for instance if mapping the stream decoder to one processing element performs worse (in every aspect) than a mapping to another processing element.

The alternatives operator satisfies the following properties, which are all known from set theory.

Proposition 9 *Let $\mathcal{C}, \mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ be arbitrary configuration sets.*

$\mathcal{C} \cup \mathcal{C} = \mathcal{C}$	(IDEMPOTENCY)
$\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}_2 \cup \mathcal{C}_1$	(COMMUTATIVITY)
$\mathcal{C}_1 \cup (\mathcal{C}_2 \cup \mathcal{C}_3) = (\mathcal{C}_1 \cup \mathcal{C}_2) \cup \mathcal{C}_3$	(ASSOCIATIVITY)
$\mathcal{C}_1 \times (\mathcal{C}_2 \cup \mathcal{C}_3) = (\mathcal{C}_1 \times \mathcal{C}_2) \cup (\mathcal{C}_1 \times \mathcal{C}_3)$	(DISTRIBUTIVITY)
$(\mathcal{C}_1 \cup \mathcal{C}_2) \times \mathcal{C}_3 = (\mathcal{C}_1 \times \mathcal{C}_3) \cup (\mathcal{C}_2 \times \mathcal{C}_3)$	(DISTRIBUTIVITY)
$\mathcal{C} \cup \varnothing = \varnothing \cup \mathcal{C} = \mathcal{C}$	(IDENTITY ELEMENT)

6.4 Constraints

When we have a set of configurations, some of which are invalid because of additional constraints, then we can apply the constraint to filter out only those configurations that satisfy the constraint. Such constraints can be expressed as a set $\mathcal{D}$ of acceptable configurations or equivalently as a proposition on the configuration space, identifying the acceptable configurations. Given a set of configurations, application of the constraint $\mathcal{D}$ amounts to taking the intersection of both sets which leaves those configurations of $\mathcal{C}$ satisfying the constraint.

Definition 15 (CONSTRAINT) *Let $\mathcal{C}$ and $\mathcal{D}$ both be sets of configurations of configuration space $\mathcal{S}$. Then $\mathcal{C} \cap \mathcal{D}$ is a set of configurations of $\mathcal{S}$ called $\mathcal{C}$ constrained to $\mathcal{D}$.*

There is however one significant problem with this operation. The essential congruence property is not satisfied. It is not the case in general that $\mathcal{C} \cap \mathcal{D} \equiv \min(\mathcal{C}) \cap \mathcal{D}$. For suppose that $\mathcal{C} = \{sb\_high, sb\_med, sb\_low\}$ models the available bit rate of the video server, and we would like to use $\mathcal{D} = \{sb\_med\}$ to express that the current bit rate that can be supported is not more than the medium level (for example because the available network bandwidth is only nw_med). In this case, $\mathcal{C} \cap \mathcal{D} = \{sb\_med\}$ and since $\min(\mathcal{C}) = \{sb\_low\}$, $\min(\mathcal{C}) \cap \mathcal{D} = \varnothing$, which means that $\mathcal{C} \cap \mathcal{D}$ and $\min(\mathcal{C}) \cap \mathcal{D}$ are obviously not equivalent. The fact that $\min(\mathcal{C}) \cap \mathcal{D} = \varnothing$ suggests that no feasible solution remains, whereas obviously there are two remaining feasible solutions. Therefore minimisation before applying a constraint may result in loss

of feasible solutions and such a constraint operator would not be suitable for a framework based on Pareto minimal sets. To remedy this problem, we have to take a closer look at constraints. By minimisation, we have removed configurations that were dominated by another configuration. Therefore, it should never be the case that the dominated configuration passes the constraint while the dominating configuration fails the test. In the example, the constraint should have been formulated as $\mathcal{D} = \{sb\_low, sb\_med\}$, expressing that also the low bit rate level is available. Then, $\min(\mathcal{C}) = \min(\mathcal{C}) \cap \mathcal{D} = \{sb\_low\}$, expressing that the low bit rate solution is the optimal solution given constraint $\mathcal{D}$ (which is intuitively correct considering only the server point of view). We concentrate on a type of constraints that satisfy the property that configurations dominating any configuration satisfying a constraint, also satisfy the constraint.

Definition 16 (SAFENESS) *A set $\mathcal{C}$ of configurations from configuration space $\mathcal{S}$ is called* safe *iff for all $\bar{c}_1, \bar{c}_2 \in \mathcal{S}$, such that $\bar{c}_1 \preceq \bar{c}_2$, $\bar{c}_2 \in \mathcal{C}$ implies that $\bar{c}_1 \in \mathcal{C}$. A safe set of configurations is also called a* safe constraint.

A safe constraint is closed under addition of dominating configurations. If a configuration is acceptable, then any configuration that dominates it must be acceptable too. This restriction is quite natural and says that there is no such thing as a configuration that is 'too good'. It is usually satisfied in practice.

For example, when two tasks are mapped onto a single processor, then the sum of the loads λ_1 and λ_2 cannot exceed the capacity p of the processor; i.e., $\lambda_1 + \lambda_2 \leq p$. It is easy to see that such a constraint is safe. If one combination fits and we reduce the load of some task, then the new combination still fits. A constraint in the stream example is that the bit rate of the stream should fit with the available network bandwidth, or that the server and application need to select the same quality setting of the stream to be able to operate together.

Proposition 10 (DOMINANCE PRESERVATION) *Let $\mathcal{C}' \preceq \mathcal{C}$ and let $\mathcal{D}$ be a safe constraint. Then, $\mathcal{C}' \cap \mathcal{D} \preceq \mathcal{C} \cap \mathcal{D}$.*

PROOF Let $\bar{c} \in \mathcal{C} \cap \mathcal{D}$. $\mathcal{C}' \preceq \mathcal{C}$, so there is some $\bar{c}' \in \mathcal{C}'$ such that $\bar{c}' \preceq \bar{c}$. Because $\mathcal{D}$ is safe, $\bar{c}' \in \mathcal{D}$ and thus $\bar{c}' \in \mathcal{C}' \cap \mathcal{D}$. □

Corollary 10 (CONGRUENCE) *If $\mathcal{C}' \equiv \mathcal{C}$ and $\mathcal{D}$ is a safe constraint, then $\mathcal{C}' \cap \mathcal{D} \equiv \mathcal{C} \cap \mathcal{D}$.*

Corollary 11 (OPERAND MINIMISATION) *If $\mathcal{D}$ is a safe constraint, then $\mathcal{C} \cap \mathcal{D} \equiv \min(\mathcal{C}) \cap \mathcal{D}$.*

In fact, a safe constraint also preserves minimality.

Proposition 11 (MINIMALITY PRESERVATION) *If $\mathcal{D}$ is a safe constraint, then* $\min(\mathcal{C} \cap \mathcal{D}) = \min(\mathcal{C}) \cap \mathcal{D}$.

PROOF ($\subseteq$) Let $\bar{c} \in \min(\mathcal{C} \cap \mathcal{D})$. Using Corollary 4 it follows that $\min(\mathcal{C} \cap \mathcal{D}) \subseteq \mathcal{C} \cap \mathcal{D} \subseteq \mathcal{D}$. Thus $\bar{c} \in \mathcal{D}$ and also $\min(\mathcal{C} \cap \mathcal{D}) \subseteq \mathcal{C} \cap \mathcal{D} \subseteq \mathcal{C}$ and $\bar{c} \in \mathcal{C}$. It remains to be shown that $\bar{c} \in \min(\mathcal{C})$. Assume the contrary, $\bar{c} \notin \min(\mathcal{C})$. It follows that there is some $\bar{c}' \in \min(\mathcal{C})$ such that $\bar{c}' \prec \bar{c}$. From safety of $\mathcal{D}$ and $\bar{c} \in \mathcal{D}$ however, it follows that $\bar{c}' \in \mathcal{C} \cap \mathcal{D}$, which contradicts the fact that $\bar{c} \in \min(\mathcal{C} \cap \mathcal{D})$.
($\supseteq$) Let $\bar{c} \in \min(\mathcal{C}) \cap \mathcal{D}$. Then $\bar{c} \in \min(\mathcal{C})$ and $\bar{c} \in \mathcal{D}$. Assume that $\bar{c} \notin \min(\mathcal{C} \cap \mathcal{D})$. Because of Corollary 4, $\bar{c} \in \mathcal{C}$ and $\bar{c} \in \mathcal{C} \cap \mathcal{D}$. Then there is some $\bar{c}' \in \min(\mathcal{C} \cap \mathcal{D})$ such that $\bar{c}' \prec \bar{c}$. But then $\bar{c}' \in \mathcal{C}$ contradicts the fact that $\bar{c} \in \min(\mathcal{C})$. Thus, the assumption is wrong and $\bar{c} \in \min(\mathcal{C} \cap \mathcal{D})$. □

If we consider arbitrary (potentially non-safe) constraints, then the constraint operator is just set intersection. We can then derive the expected straightforward properties such as idempotency, commutativity, associativity, and various distribution properties with respect to free product and alternatives. However, since only safe constraints are of practical interest, it is more interesting to consider properties for safe constraints. Technically speaking, safe constraints are just sets of configurations, but in practice they are not very meaningful as such because of the safeness assumption, which essentially assumes that all possible configurations in a configuration space are feasible and no selection of configurations is possible. This means, for example, that although set intersection is idempotent, this property is not meaningful in practice. Similarly, there is also no meaningful commutativity property. The following proposition summarises the meaningful properties of safe constraints. Note that if $\mathcal{D}_1$ and $\mathcal{D}_2$ are safe constraints, then both $\mathcal{D}_1 \cup \mathcal{D}_2$ and $\mathcal{D}_1 \cap \mathcal{D}_2$ are safe constraints.

Proposition 12 *Let $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_2$ be arbitrary sets of configurations (i.e., not necessarily safe) and let $\mathcal{D}$, $\mathcal{D}_1$, $\mathcal{D}_2$ be safe constraints.*

$$\mathcal{C} \cap (\mathcal{D}_1 \cap \mathcal{D}_2) = (\mathcal{C} \cap \mathcal{D}_1) \cap \mathcal{D}_2 \qquad \text{(ASSOCIATIVITY)}$$
$$(\mathcal{C}_1 \cup \mathcal{C}_2) \cap \mathcal{D} = (\mathcal{C}_1 \cap \mathcal{D}) \cup (\mathcal{C}_2 \cap \mathcal{D}) \qquad \text{(DISTRIBUTIVITY)}$$
$$\mathcal{C} \cap (\mathcal{D}_1 \cup \mathcal{D}_2) = (\mathcal{C} \cap \mathcal{D}_1) \cup (\mathcal{C} \cap \mathcal{D}_2) \qquad \text{(DISTRIBUTIVITY)}$$
$$\mathcal{C} \cap \varnothing = \varnothing \qquad \text{(ZERO ELEMENT)}$$

PROOF Straightforward from set theory. □

The associativity property states that one can apply two constraints after each other (right-hand side of the equality) or the combined constraint (left-hand side) with the same result. Note that $\mathcal{C} \cap \mathcal{D}$ is in general not safe, so we do not have an associativity property if we replace $\mathcal{D}_1$ in the above associativity property by an arbitrary set of configurations $\mathcal{C}$. Also, because $\mathcal{C} \times \mathcal{D}$, $\mathcal{D} \times \mathcal{C}$, $\mathcal{C} \cup \mathcal{D}$, and $\mathcal{D} \cup \mathcal{C}$ are in general not safe, we only have two meaningful distribution properties, both with the alternatives operator. The first one states that subsets of configurations can be constrained separately. Note that union of constraints can be seen as relaxation of constraints. The second property therefore states that applying a relaxed constraint $\mathcal{D}_1 \cup \mathcal{D}_2$ to configuration set $\mathcal{C}$ corresponds to combining the sets obtained by constraining $\mathcal{C}$ separately by $\mathcal{D}_1$ and $\mathcal{D}_2$. The final property states that the constraint excluding any configuration, namely the empty set, acts as a zero element.

Because $\min(\mathcal{D})$ is safe, one could investigate properties concerning the interaction between minimisation and safe constraints. However, the result of $\min(\mathcal{D})$ is usually not a very meaningful constraint in practice because it is too restrictive.

An interesting observation is that an arbitrary set of configurations can be turned into a safe one.

Definition 17 (SAFETY CLOSURE) *If $\mathcal{C}$ is a set of configurations, then $\mathcal{C}^{sc}$ is the smallest safe set of configurations which contains $\mathcal{C}$.*

Note that the closure always exists if the configurations are well-ordered. Safety closure is not only a convenient operation, because it allows to provide only partial constraint specifications, which can then be closed to obtain a safe constraint, but it also gives the following distribution laws with the free product operator.

Proposition 13 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be arbitrary sets of configurations and $\mathcal{D}$ a safe constraint.*

$$(\mathcal{C}_1 \cap \mathcal{D}) \times \mathcal{C}_2 = (\mathcal{C}_1 \times \mathcal{C}_2) \cap (\mathcal{D} \times \mathcal{C}_2)^{sc} \qquad \text{(DISTRIBUTIVITY)}$$
$$\mathcal{C}_1 \times (\mathcal{C}_2 \cap \mathcal{D}) = (\mathcal{C}_1 \times \mathcal{C}_2) \cap (\mathcal{C}_1 \times \mathcal{D})^{sc} \qquad \text{(DISTRIBUTIVITY)}$$

PROOF The first equation is proved from right to left as follows. $(\mathcal{C}_1 \times \mathcal{C}_2) \cap (\mathcal{D} \times \mathcal{C}_2)^{sc} = (\mathcal{C}_1 \times \mathcal{C}_2) \cap (\mathcal{D} \times \mathcal{C}_2^{sc}) = (\mathcal{C}_1 \cap \mathcal{D}) \times (\mathcal{C}_2 \cap \mathcal{C}_2^{sc}) = (\mathcal{C}_1 \cap \mathcal{D}) \times \mathcal{C}_2$. The second equation is symmetrical, □

In these two properties, the new constraints $(\mathcal{D} \times \mathcal{C}_2)^{sc}$ and $(\mathcal{C}_1 \times \mathcal{D})^{sc}$ do not constrain the quantities that are newly added through $\mathcal{C}_2$ and $\mathcal{C}_1$ respectively. In fact, these two properties are good examples of the above mentioned partial constraint specification. The original constraint $\mathcal{D}$ constrains in both cases only part of the configuration space.

Finally, one could wonder whether $\mathcal{C}_1 \cup (\mathcal{C}_2 \cap \mathcal{D}) = (\mathcal{C}_1 \cup \mathcal{C}_2) \cap (\mathcal{C}_1 \cup \mathcal{D})^{sc}$. It is an interesting exercise to verify that this is not the case.

6.5 Abstraction

Sometimes a configuration space contains too much information; some quantities are not (or no longer) relevant. Abstraction can then be used to remove such a dimension from the configuration space. To be precise, and for technical convenience, abstraction of a quantity is defined based on the index of the quantity in the configuration space.

Definition 18 *If $\bar{a} = (a_1, \ldots, a_n)$ is a tuple of length n and $1 \leq k \leq n$, then $\bar{a} \downarrow k$ denotes $(a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n)$. Moreover, $\downarrow$ is lifted to sets of tuples as follows: $A \downarrow k = \{\bar{a} \downarrow k \mid \bar{a} \in A\}$.*

Definition 19 (ABSTRACTION) *Let $\mathcal{C}$ be a set of configurations of configuration space $\mathcal{S} = Q_1 \times Q_2 \times \ldots \times Q_n$. Then, $\mathcal{C} \downarrow k$ is a set of configurations over configuration space $\mathcal{S} \downarrow k = Q_1 \times \ldots \times Q_{k-1} \times Q_{k+1} \times \ldots \times Q_n$.*

An example of using the abstraction operator could be the case where the server, the wireless network and the hand-held application have been combined and we have enforced a constraint that the hand-held and the server will use the same mode of video encoding. After this, the actual mode is no longer relevant. It can be abstracted.

Proposition 14 (DOMINANCE PRESERVATION) *If $\mathcal{C}' \preceq \mathcal{C}$, then $\mathcal{C}' \downarrow k \preceq \mathcal{C} \downarrow k$.*

PROOF Let $\bar{c} \in \mathcal{C} \downarrow k$ and $\bar{c}_1 \in \mathcal{C}$ such that $\bar{c}_1 \downarrow k = \bar{c}$. There is some $\bar{c}'_1 \in \mathcal{C}'$ such that $\bar{c}'_1 \preceq \bar{c}_1$. Then $\bar{c}'_1 \downarrow k \preceq \bar{c}$ and $\bar{c}'_1 \downarrow k \in \mathcal{C}' \downarrow k$. □

gThis proposition is of the same form as property (4). Hence, the following corollaries follow.

Corollary 12 (CONGRUENCE) *If $\mathcal{C}' \equiv \mathcal{C}$, then $\mathcal{C}' \downarrow k \equiv \mathcal{C} \downarrow k$.*

Corollary 13 (OPERAND MINIMISATION) $\mathcal{C} \downarrow k \equiv \min(\mathcal{C}) \downarrow k$.

Note however, that a Pareto minimal set may no longer be Pareto minimal after abstraction. Property (2) does not hold. It is not the case, in general, that $\min(\mathcal{C} \downarrow k) = \min(\mathcal{C}) \downarrow k$. In the example sketched above, as long as the (unrelated) encoding modes are part of the configuration, a configuration in mode A cannot dominate a configuration in mode B. When we abstract from the mode, we can choose the best configuration amongst all video modes and a configuration in mode A can dominate a configuration in mode B. Therefore, minimisation may be required after abstraction.

The abstraction operator satisfies the following properties. Note that abstracting away a quantity in a safe constraint preserves safety.

Proposition 15 *Let $\mathcal{C}$, $\mathcal{C}_1$, and $\mathcal{C}'_1$ be configuration sets and $\mathcal{D}$ a safe constraint from configuration space $Q_1 \times \ldots \times Q_m$, and $\mathcal{C}_2$ a configuration set from space $Q_{m+1} \times \ldots \times Q_n$, with $1 \leq m < n$. Let k and k' with $k > k'$ be in $\{1, \ldots, m\}$, and l in $\{m+1, \ldots, n\}$.*

$$(\mathcal{C} \downarrow k) \downarrow k' = (\mathcal{C} \downarrow k') \downarrow (k-1)$$
$$(\mathcal{C}_1 \times \mathcal{C}_2) \downarrow k = (\mathcal{C}_1 \downarrow k) \times \mathcal{C}_2$$
$$(\mathcal{C}_1 \times \mathcal{C}_2) \downarrow l = \mathcal{C}_1 \times (\mathcal{C}_2 \downarrow (l-m))$$
$$(\mathcal{C}_1 \cup \mathcal{C}'_1) \downarrow k = (\mathcal{C}_1 \downarrow k) \cup (\mathcal{C}'_1 \downarrow k) \qquad \text{(DISTRIBUTIVITY)}$$
$$(\mathcal{C} \cap \mathcal{D}) \downarrow k \subseteq (\mathcal{C} \downarrow k) \cap (\mathcal{D} \downarrow k)$$

PROOF Straightforward. □

The first property states that the order of abstracting away multiple quantities is irrelevant (when correcting for changes in the index number of quantities due to abstraction). Based on this property, we may unambiguously write $\mathcal{C} \downarrow S$, with S a subset of the index set $\{1, \ldots, m\}$, to denote the simultaneous abstraction from multiple quantities. The second and third property show that abstraction of a certain quantity can equivalently be performed before or after applying a free product. The fourth property states the distribution of abstraction over alternatives. In general, abstraction does not distribute over the constraint operator, but the fifth property above shows that applying a safe constraint before abstraction yields a more constrained set than applying abstraction before constraining the configuration set. Abstraction might in fact relax a constraint.

6.6 Overview

In summary, Table 1 gives an overview of the operators of the algebra and their properties, focussing on support for intermediate operand minimisation (congruence) and preservation of minimality, which allows minimisation to be performed before applying the operator without the need to minimise again afterwards. If we restrict ourselves to safe constraints, then all operators support intermediate minimisation, and hence, the Pareto approach.

operator	*allows operand minimisation*	*preserves minimality*
minimisation	+	+
free product	+	+
alternatives	+	-
constraints	-	-
safe constraints	+	+
abstraction	+	-

Table 1: Properties of the operators

7 Composition of the Operators

Using the basic operators of the algebra, we can build more sophisticated compositions.

7.1 Producer-Consumer

Using the previously defined operators, we can compute configurations of a combination of producing and consuming entities. Assume we interpret one dimension of one configuration space as a produced quantity, for instance, the amount of computation power of a processor in the hand-held platform of our example. We can match that to another quantity of another configuration space, which plays the role of the consumer of that quantity. For instance, the video codec application running on the hand-held, requires a certain amount of computational effort in a certain configuration. For the platform, producing more computational power is better. For the codec, it is the other way around. The less computational power it requires, the better it is. The relationship between the two should be constrained such that the platform produces at least the amount of computational effort required by the codec.

This can be realised using a combination of a free product, to combine producer and consumer to a single system, and a constraint to guarantee that the produced and consumed quantities match. This constraint can be expressed with a mapping function between the producing and the consuming quantities, that is monotonically decreasing. For instance, if the consumed quantity is computational effort expressed for instance in MFlops (millions of floating point operations), and the produced quantity is computational power in 1/MFlops, then this mapping is $m \mapsto \frac{1}{m}$. Note the difference between computational power and computational effort to ensure that smaller is better.

Let $\mathcal{PC} = \mathcal{P} \times \mathcal{C}$ be the combined configuration space of producer $\mathcal{P}$ and consumer $\mathcal{C}$ and $f : Q_P \to Q_C$ the monotonically decreasing mapping between produced quantity Q_P and consumed quantity Q_C. The constraint $\mathcal{M}$ is expressed as follows:

$$\mathcal{M} = \{\bar{c} \in \mathcal{PC} \mid \bar{c}(Q_C) \preceq_{Q_C} f(\bar{c}(Q_P))\}.$$

Note that the constraint $\mathcal{M}$ is indeed safe, because f is monotonically decreasing. The producer-consumer system is then

$$\mathcal{PC} \cap \mathcal{M}.$$

Next, we may want to abstract from the quantities that were matched: Q_P and Q_C. Assume their index numbers in space $\mathcal{PC}$ are p and c, respectively. The producer-consumer operation can then be expressed as follows:

$$(\mathcal{PC} \cap \mathcal{M}) \downarrow \{p, c\}.$$

7.2 Join

Another useful construct is a join operation, as used in relational databases. The operation can be expressed using the free product and a constraint, the latter stating that certain quantity values be related. A direct computation of the result along the lines of these operators would first construct the entire product space and then discard a lot of configurations where the quantities do not match. It can be computed much more

efficiently directly, however. For instance, the server and decoder application in our example need to agree on the quality level of the stream they are communicating.

7.3 The Example Revisited

With the algebra introduced in this paper we continue the example of the wireless video stream system. The hand-held application decodes the video stream. It cannot handle the high quality stream associated with mode C, but can choose between medium and low quality modes B and A. Related to that is the computational effort required for the decoding. $AEffort = \mathbb{N}$, measured, say, in MFlops and obtained by profiling or static analysis. The two configurations of the application are mode A for 10 MFlops, mode B for 50:

$$\{(A, 10), (B, 50)\}.$$

The computation effort is provided by the hand-held platform. The platform can deliver more effort at the expense of a higher energy usage. The two quantities involved are platform effort and power dissipation. $PEffort$ is a number in 1/MFlops. The corresponding power consumption is given in a quantity we call $PPower$, a real number in Watt. The valid platform configurations, showing the trade-off between performance and power consumption, are

$$\{(1/12, 0.23), (1/33, 0.8), (1/75, 1.2)\}.$$

The application runs on the platform and the computation efforts have to be matched (via a producer-consumer operation). Applying the free product and the constraint of the producer-consumer operation leads to the following valid configurations:

$$\{(A, 10, 1/12, 0.23), (A, 10, 1/33, 0.8), (A, 10, 1/75, 1.2),$$
$$(B, 50, 1/75, 1.2)\}.$$

Note that this set is minimal. Once application and platform have been connected, the computational efforts are not interesting anymore. After abstraction, this gives the configurations showing the video (en)coding mode and corresponding power consumption:

$$\{(A, 0.23), (A, 0.8), (A, 1.2), (B, 1.2)\}$$

in configuration space $EncodingMode \times PPower$.

This set of configurations is not minimal. There are three ways to decode mode A, but only the most energy-efficient one is interesting. After minimisation we get:

$$\{(A, 0.23), (B, 1.2)\}.$$

Finally, we have to combine the hand-held with the network and server with the constraint that the video modes in the server and the hand-held are matched (a join) and the required bit rate and available network bandwidth are matched (producer-consumer). Assume that medium network bandwidth (nw_med) is available. (This excludes the high bit rate video mode, but this mode is already excluded by the hand-held as well.) It is not difficult to see that the result is the following two configurations:

$$\{(A, sq\_low, sb\_low, nw\_med, A, 0.23),$$
$$(B, sq\_med, sb\_med, nw\_med, B, 1.2)\}.$$

Abstracting now irrelevant information yields:

$$\{(sq\_low, sb\_low, 0.23), (sq\_med, sb\_med, 1.2)\}.$$

The conclusion of the exercise is that we can select a low quality stream, having low bandwidth usage on the network and low power consumption in the hand-held of 0.23W. The alternative offers medium quality video, which uses more bandwidth of the network and 1.2W of power on the hand-held. If we prioritise quality over power consumption and finally bandwidth, the optimal configuration becomes: $(sq\_med, sb\_med, 1.2)$. Other priorities could lead to the other configuration being preferred. Priorities could also differ over time or depend on system parameters such as remaining battery power. A different configuration may become the better option and a system reconfiguration could be initiated.

8 MPEG-4 Case-Study

In this section, we have a closer look at the MPEG-4 streaming case-study. We consider the individual components in more detail than the abstract version used in the previous sections. We derive Pareto models of

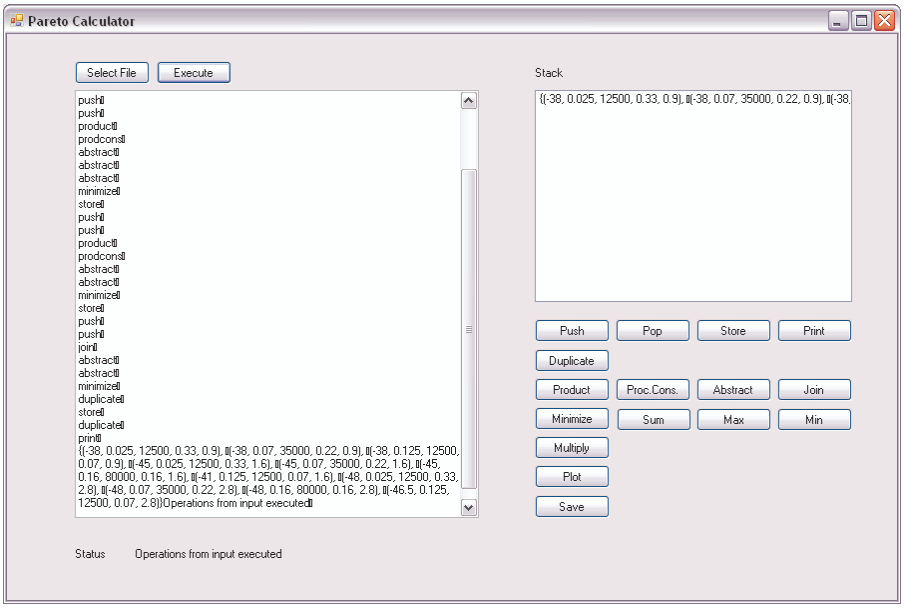


Figure 6: The Pareto Calculator

the encoder, wireless transmission, decoder and processing platform and subsequently compute the trade-offs of the composition of the components into the entire delivery chain. The corresponding operations have been computed using a prototype implementation of the algebraic operators in the form of a Pareto calculator (see Figure 6). For most of the quantities of the components the ordering is straightforward and is not mentioned explicitly. In particular, the natural way to express some quantities is such that larger is better in contrast with the general assumption used so far in this paper. We could introduce quantities where smaller is better for instance by taking the reciprocal of the numbers as we have done in previous sections, but for readability we do not do so in this section. Note also that this case-study does not intend to give an accurate analysis of MPEG-4 streaming, but rather to illustrate the possibilities of the algebra and the way it can be used in practical situations. Some of the numbers used in the component models are estimates that are not based on real measurements.

8.1 Encoder

The MPEG-4 encoder has a number of parameters that determine the properties of the output stream. We assume that the encoding takes place on a server device and resource consumption of the encoding process is not of interest and is therefore not modelled explicitly. The encoding process can then be characterised with the following attributes (although in practice there are even more.)

- The frame rate of the encoded stream, expressed in the number of frames per second (fps).
- The frame size, the number of pixels used for each of the video frames. This is typically selected from a discrete set of choices, for instance CIF (Common Intermediate Format), being 352 by 288 pixels, QCIF, a quarter of CIF or 4CIF having four times the number of pixels of CIF.
- The quantisation parameter determines the quality (amount of data) with which the individual texture blocks of the image are encoded.
- Quality of the resulting stream. When the above parameters have been set, the encoder produces a stream of video data. The quality of that stream depends on the parameter settings. There is no known good measure of quality for moving picture images. It may even need to be considered a multi-dimensional attribute, being good in either capturing quick movement accurately, or displaying static textures in high quality. An often used measure is (average) PSNR (Peak Signal-to-Noise

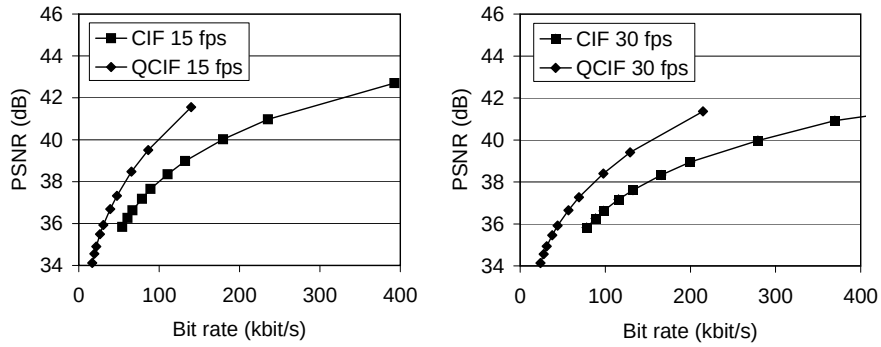


Figure 7: Profiled quality and bit rate for the encoder [3]

Parameter settings	*Bit rate (kbit/s)*	*Quality*
S_1 (QCIF, 15 fps)	64	38.0
S_2 (QCIF, 15 fps)	384	45.0
S_3 (QCIF, 30 fps)	64	41.0
S_4 (QCIF, 30 fps)	384	48.0
S_5 (CIF, 15 fps)	64	46.5
S_6 (CIF, 15 fps)	384	52.5
S_7 (CIF, 30 fps)	64	48.5
S_8 (CIF, 30 fps)	384	56.0

Table 2: MPEG-4 encoder settings and corresponding bit rate and quality

Ratio), although this does not perfectly correspond to quality as perceived by actual human viewers. Moreover, it does not capture the impact of spatial and temporal resolution. For simplicity, we use in this case-study a rather artificial (real) number representing quality, based on average PSNR, temporal resolution and spatial resolution. We take the PSNR in dB as a basis and add 5 when the stream uses a frame rate of 30 fps instead of 15, and we add 10 when the stream uses CIF instead of QCIF. It is not difficult to change or extend this to better quality models.

- Bit rate of the resulting stream. Similar to the quality, the parameter settings also lead to a stream with a particular bit rate. Generally, a higher quality requires a higher bit rate, but higher bit rates may be more costly or impossible to communicate. Hence, there is a trade-off between the two.

The set of configurations of the encoder is constructed as follows. We start with the encoder parameters that can be set. In practice that would be a space consisting of three, unordered quantities, frame rate, frame size and quantisation parameter. Each of these quantities is unordered, because we are not so much interested in the parameter settings themselves, but rather in their impact on bit rate and quality. The configuration space of these three quantities can also be interpreted as a single quantity itself. To keep things manageable in our example, we do this and call this quantity $MPEGParams$.

From off-line profiling of the MPEG-4 encoder, bit rate and PSNR picture quality can be determined as a function of the parameter settings. Using the technique of derived quantities of Section 5.3, we extend the configuration set of parameter settings with both quantities $MPEGQuality$ and $MPEGBitRate$. Note that both profiled functions are trivially monotone, since the domain, the parameter settings, is unordered. The profile information is taken from [3], as depicted in Figure 7 (which is in fact accurate only for a particular type of video sequence). The selected configurations include a choice between CIF and QCIF spatial resolutions, between 15fps and 30fps temporal resolutions and corresponding quantisation parameter such that the bit rate equals approximately 64 kbit/s or 384 kbit/s. The configuration set $Encoder$ contains the eight discrete configurations that are listed in Table 2. Note that no reduction is possible by minimisation, because the MPEG-4 parameter settings are unordered. They are not abstracted, because the choice of settings may still impact the decoder, which needs to match the same settings.

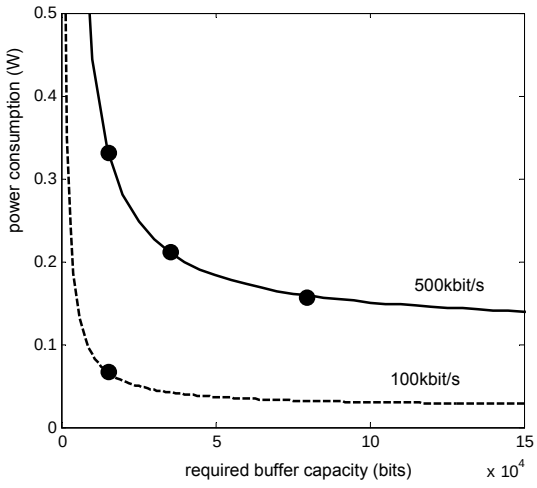


Figure 8: Trade-off between buffer capacity ($\sim$ latency), power consumption and bit rate

8.2 Wireless Transmission

Next, we consider the wireless transmission of the video data, with particular attention to the power consumed in the radio transceiver at the hand-held side of the communication and a strategy to conserve energy. To save energy, data from the stream can be sent across the network in bursts. During the time between bursts, the radio transceiver can be shut down, thus saving energy. The longer the period, the more energy can be saved, but also, the more buffering capacity is needed to store stream data and the more latency is introduced by the process. The parameters that play a role in this component are the following.

- The average bit rate of the stream that needs to be transported is r bits per second.
- The net bandwidth, when the network is used at full speed, is BW bits per second. This may depend on environmental conditions, distance between sender and receiver, and so forth.
- The power used by the transceiver while actively transmitting is P_{on} Watt.
- Each burst of data sent on the network involves an overhead of τ seconds transmitting protocol information at full power but no data of the actual stream.
- The time period between the start of successive bursts is T seconds. T is a parameter that can be controlled for optimisation.

The required duty cycle of the transceiver to obtain an average bit rate of r equals r/BW, so during each period of length T, the transceiver is on for $T \cdot r/BW$ seconds, plus an overhead of τ seconds. The energy, to transmit data during n bursts with period T is then as follows: $n \cdot P_{on} \cdot (\tau + T \cdot r/BW)$. Assuming that power consumption is negligible when the transceiver is switched off, the average transmission power is then

$$P = P_{on} \left(\frac{\tau}{T} + \frac{r}{BW} \right).$$

From this, it is obvious that a longer period T leads to a lower average transmission power and that availability of more bandwidth reduces power as well.

The buffers required to store the bursts of data at the sender and at the receiver side need to be equally large and sufficient to contain approximately one entire burst (of $T \cdot r$ bits). More precisely, by the end of the burst, all data should be available, so the buffer needs to be able to hold all data that arrives until the burst begins. This leads to the following approximation of the buffer size.

$$(1 - \frac{r}{BW}) \cdot T \cdot r \approx T \cdot r.$$

Average power (W)	*Latency (s)*	*Buffer capacity (bits)*	*Bit rate (kbit/s)*
0.07	0.125	12500	100
0.33	0.025	12500	500
0.22	0.07	35000	500
0.16	0.16	80000	500

Table 3: Wireless transmission configurations

The approximation is only valid if the bit rate of the stream is much smaller than the network bandwidth. We assume that only the buffers at the receiving hand-held side are relevant as the server has no problem reserving sufficient buffer capacity. Because of the buffering, also the latency of the transmission increases. Straightforward analysis of the transmission component tells us that the time between arrival of the first information in the input buffer and on the other side of the channel is:

$$(1 - \frac{r}{BW}) \cdot T \approx T.$$

As a consequence, we have a trade-off between latency, buffer size, transmission power and bit rate, depending on the bandwidth provided by the network, using the period T as a controllable parameter.

$$\left(T, T \cdot r, P_{on}\left(\frac{\tau}{T} + \frac{r}{BW}\right), r\right).$$

For all T, the corresponding point is a Pareto point. We assume the following actual numbers for the parameters. When on, the wireless adapter consumes $P_{on} = 1.3$W, when the transceiver is off, no power is consumed. We analyse the situation for two distinct bit rates of 100 kbit/s and 500 kbit/s respectively. We assume that the overhead time τ equals 5ms and that the net bandwidth of the network under current conditions is 5.5 Mb/s. Then, the average power consumption becomes (for 100 kbit/s)

$$P = 1.3 \cdot (\frac{0.005}{T} + \frac{100 \cdot 10^3}{5.5 \cdot 10^6}) = \frac{0.0065}{T} + 0.024.$$

In Figure 8, the trade-off between buffer capacity, latency, power consumption and bit rate is depicted, using bit rates of 100 kbit/s and 500 kbit/s. In practice, a reasonable approximation of all the possible configuration points may be obtained from selecting a number of concrete configuration points, for instance the black dots in the figure. For the 100 kbit/s stream, one point may suffice, while for the 500 kbit/s stream with a more gradual slope, three interesting points are identified. We work with these four points in this case-study. Note that the lower-left point may seem to dominate the others in this graph, but this is not the case, since they have higher bit rates. The latency is also not explicitly depicted in this figure, because it is proportional to the required buffer capacity (the proportion being the bit rate).

From the presented analytical model of the wireless transmission, we have derived a set of configurations with the following quantities: Average transmission power $TransmPower$, latency $TransmLatency$, buffer capacity $TransmBufferCapacity$, and bit rate $TransmBitRate$. Table 3 shows the configurations we have used. This is the set $Transmission$ of configurations.

8.3 Decoder

The MPEG-4 decoder in our example is a software decoder running on a platform which is modelled independently. The same parameters that the encoder can set to adjust the video stream also play a role in the decoding of the video stream. Similarly, the same quality and bit rate profiles apply, so there is no need to model them again for this component. An attribute that does play a role for the decoder is the amount of effort that is needed from the platform processor to perform the decoding. Similar to the encoder attributes, we assume that the decoder has also been profiled off-line on the processor (type) of the platform it runs on and the profiling results are captured by a function giving the number of instructions per second required from the processor in MIPS (millions of instructions per second). Although other aspects such as memory

Parameter settings	*Required computational effort (MIPS)*
S_1	97
S_2	156
S_3	192
S_4	303
S_5	378
S_6	545
S_7	749
S_8	1100

Table 4: MPEG-4 decoder settings and corresponding required computational effort

Operating mode	*Available computational effort (MIPS)*	*Average power consumption (W)*
M_1	400	2.8
M_2	200	1.6
M_3	100	0.9

Table 5: Processor modes with performance and power consumption

usage or decoding latency could also be relevant, we refrain from modelling them in this simple example. Similar to the encoder model, the configuration set is generated by starting with the decoder parameters and adding the profile information as a derived quantity $DecoderCompEffort$. The decoder is captured in the following configuration space

$$MPEGParams \times DecoderCompEffort.$$

The configurations of the set $Decoder$ are given in Table 4.

8.4 Computational Resources

The final component of the delivery chain is the platform on which the decoder runs. We assume a simple model of a processor with dynamic voltage and frequency scaling possibilities. The processor has a finite number of operating modes, each with a specific voltage and frequency, power consumption and performance in millions of instructions per second. More accurate models can be made, differentiating different kinds of instructions, including memory accesses, including energy consumption of other peripherals such as a memory controller. We use the simple model for our illustrative purposes.

The processor platform data is modelled in the following configuration space:

$$ProcessorMode \times ProcessorCompEffort \times ProcessorPower.$$

Note that here, as in some of the examples before, the quantity $ProcessorCompEffort$ is ordered in the opposite way (larger is better) compared to $DecoderCompEffort$, where smaller is better. $ProcessorPower$ captures the average power consumed by the processor in the particular mode. The configurations of the set $Processor$ are given in Table 5.

8.5 End-to-end Delivery Chain Trade-offs

Each of the component models described above can be obtained off-line, at design time by profiling or by analytical methods. The dynamic and ad-hoc nature of wireless networked components and their environments makes it impossible to analyse every conceivable situation at design time. Therefore, it is necessary to determine at run-time what kind of optimisation the specific configuration of components allows. The algebra of Pareto points can be used to study the composition of the components into an end-to-end delivery chain. The analysis follows the line of reasoning of the example in Section 7.3.

Average processor power consumption (W)	*MPEG-4 parameter settings*
0.9	S_1
1.6	S_2
1.6	S_3
2.8	S_4
2.8	S_5

Table 6: Processor power consumption vs. MPEG-4 parameter settings

First, we put the software decoder and the platform together to see what kind of video the hand-held can handle at what cost. Here, it is important to check that the platform delivers the required computational effort for decoding at particular parameter settings, and at which cost in terms of power consumption.

We take the product $Processor \times Decoder$ and subsequently apply a producer-consumer constraint on the quantities $ProcessorCompEffort$ and $DecoderCompEffort$, which ensures that the decoder can run on the processor at a particular level of power consumption. After this we abstract from both quantities as well as the $ProcessorMode$, in which we are not interested, and we minimise. The result of the composition is a set of configurations in the space:

$$ProcessorPower \times MPEGParams.$$

The configurations are shown in Table 6. Note that the parameter settings S_6, S_7 and S_8 are out of reach of the processor of the hand-held.

Second, we combine the encoder with the network transport. The set $Transmission$ contains the configurations according to the currently available network conditions. We start from $Transmission \times Encoder$ and apply a producer-consumer constraint on the $TransmBitRate$ and $MPEGBitRate$ quantities, which ensures that the transmission parameter is set in accordance with the bit rate requirements. After this, we abstract from both bit rate quantities and minimise. We obtain a set of configurations in the space

$$TransmPower \times TransmLatency \times TransmBufferCapacity \times MPEGParams \times MPEGQuality.$$

The set contains 24 configurations, shown in Table 7.

Finally, the intermediate results are put together. The processor-decoder combination and the transmission-encoder combination are combined with a join operation on the common quantity $MPEGParams$; encoder and decoder have to operate with the same stream parameters. We are no longer interested in the parameters themselves and abstract them away from the configurations. (Note that 'behind the scenes', the MPEG-4 parameter settings need to be remembered in order to select the desired configuration, but this is not relevant for illustrating the algebra.) By now we are left with a set of configurations in the space

$$ProcessorPower \times TransmPower \times TransmLatency \times TransmBufferCapacity \times MPEGQuality.$$

There are 15 configurations left (Table 8), because it is now clear that the higher quality settings cannot be supported by the hand-held. Next, we like to sum the power consumption of processor and transmission, since we do not care which one is draining our battery as long as the total is optimised. In terms of the algebra, this means that first we add the sum of $ProcessorPower$ and $TransmPower$ as a derived quantity $TotalPower$ (summation is monotone) and subsequently abstract the individual power quantities and minimise again. We end up with configurations in the space

$$TransmLatency \times TransmBufferCapacity \times MPEGQuality \times TotalPower$$

a trade-off between quality, latency, power consumption and required buffer capacity. The 11 remaining Pareto points are listed in Table 9. Figure 9 shows two views on this multi-dimensional trade-off. Figure 9(a) shows latency vs. power consumption of different configurations. Correlation between the two aspects is not very strong. Low latency does not need to cost a lot of energy, since most energy is consumed in the processor and latency is introduced with the transmission process. Figure 9(b) shows quality of the

Power (W)	*Latency (s)*	*Buffer capacity (bits)*	*MPEG parameters*	*Quality*
0.07	0.125	12500	S_1	38.0
0.33	0.025	12500	S_1	38.0
0.22	0.07	35000	S_1	38.0
0.33	0.025	12500	S_2	45.0
0.22	0.07	35000	S_2	45.0
0.16	0.16	80000	S_2	45.0
0.07	0.125	12500	S_3	41.0
0.33	0.025	12500	S_3	41.0
0.22	0.07	35000	S_3	41.0
0.33	0.025	12500	S_4	48.0
0.22	0.07	35000	S_4	48.0
0.16	0.16	80000	S_4	48.0
0.07	0.125	12500	S_5	46.5
0.33	0.025	12500	S_5	46.5
0.22	0.07	35000	S_5	46.5
0.33	0.025	12500	S_6	52.5
0.22	0.07	35000	S_6	52.5
0.16	0.16	80000	S_6	52.5
0.07	0.125	12500	S_7	48.5
0.33	0.025	12500	S_7	48.5
0.22	0.07	35000	S_7	48.5
0.33	0.025	12500	S_8	56.0
0.22	0.07	35000	S_8	56.0
0.16	0.16	80000	S_8	56.0

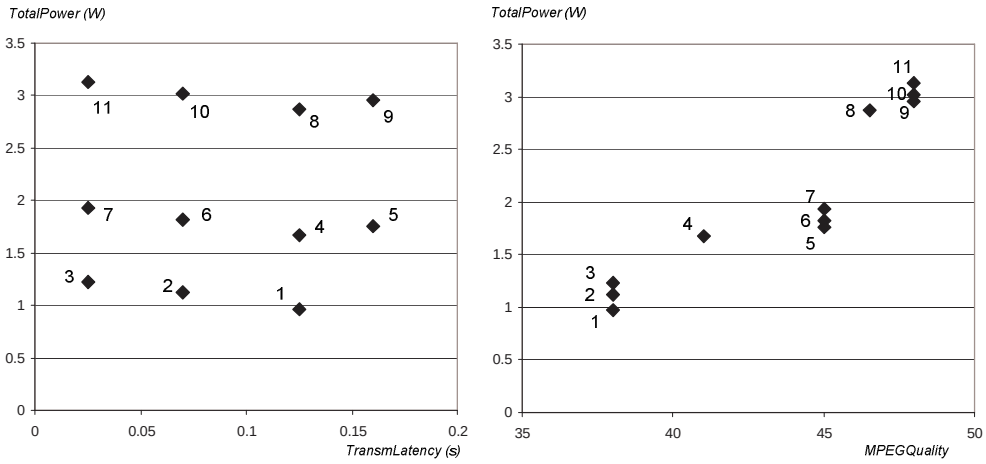
Table 7: Profile of wireless transmission and encoder combined

Processor power (W)	*Transmission power (W)*	*Latency (s)*	*Buffer capacity (bits)*	*Quality*
0.9	0.07	0.125	12500	38.0
0.9	0.33	0.025	12500	38.0
0.9	0.22	0.07	35000	38.0
1.6	0.33	0.025	12500	45.0
1.6	0.22	0.07	35000	45.0
1.6	0.16	0.16	80000	45.0
1.6	0.07	0.125	12500	41.0
1.6	0.33	0.025	12500	41.0
1.6	0.22	0.07	35000	41.0
2.8	0.33	0.025	12500	48.0
2.8	0.22	0.07	35000	48.0
2.8	0.16	0.16	80000	48.0
2.8	0.07	0.125	12500	46.5
2.8	0.33	0.025	12500	46.5
2.8	0.22	0.07	35000	46.5

Table 8: Profile of all components combined

Configuration #	*Latency (s)*	*Buffer capacity (bits)*	*Quality*	*Total power consumption (W)*
1	0.125	12500	38.0	0.97
2	0.07	35000	38.0	1.12
3	0.025	12500	38.0	1.23
4	0.125	12500	41.0	1.67
5	0.16	80000	45.0	1.76
6	0.07	35000	45.0	1.82
7	0.025	12500	45.0	1.93
8	0.125	12500	46.5	2.87
9	0.16	80000	48.0	2.96
10	0.07	35000	48.0	3.02
11	0.025	12500	48.0	3.13

Table 9: Pareto points of the MPEG-4 delivery chain



(a) (b)

Figure 9: Trade-offs of the MPEG-4 delivery chain

decoded stream vs. power consumption. Here a strong correlation can be seen and power consumption can be traded for quality.

A run-time resource and quality management system would resolve this trade-off on the basis of specific optimisation goals or strategies, which can be expressed by cost functions. It might for instance choose to give priority to video quality over power consumption under a latency constraint of 0.1s, using a corresponding prioritising cost function as discussed in Section 5.2. This leads to the selection of configuration 10. Giving priority to power consumption leads to the selection of configuration 2.

9 Computational and Complexity Issues

Pareto sets are formed by combining components and their Pareto sets with the operations of the algebra. The number of Pareto optimal configurations can grow very fast when components are added (in principle at an exponential rate). It is therefore crucial to keep the number of configurations as low as possible, through minimisation (preserving equivalence) or, if necessary, through some form of approximation. One of the advantages of the algebra is that it allows minimisation of the intermediate results of the computation. It also supports abstraction from certain dimensions and the use of cost functions that partially resolve design decisions to reduce the size of the configuration sets.

There are two general, different application scenarios for Pareto analysis and for the algebra. One is the need to compute sets of Pareto points at run-time and potentially on resource-constrained devices and the other is computing them off-line, at design-time, on a powerful computer for design-space exploration purposes. One of the crucial operations is minimisation of a set of configurations to its Pareto points. The best known algorithm is $\mathcal{O}(N(\log N)^d)$ [2, 11] where d is the number of dimensions of the configuration space and N the number of configurations. The algorithm is quite involved however and is efficient for large N in particular if the number of dimensions is high. The number of points for which the algorithm is faster than simpler algorithms such as the Simple Cull algorithm [19] having a complexity of $\mathcal{O}(N^2)$ is quite high. This makes the algorithm particularly suited for off-line analysis. Hybrid algorithms are also used in this situation [19]. For run-time analysis in resource-constrained devices, the number of configurations has to be limited and the $\mathcal{O}(N^2)$ algorithms are likely to outperform the algorithm of [2, 11]. A similar effect is observed for data structures for storing sets of configurations. The quad-tree data structure [6, 14] is shown to be more efficient than linear lists in [14], but also here, the gain is achieved for large numbers of data points. [19] shows that when keeping points lexicographically sorted, normalised linear tables can be advantageous for computing the union and intersection of sets. Pruning the set of Pareto points may be necessary if it is still too large. The challenge is then to find the best approximation of a set of Pareto points with a limited number of points [12, 21]. We have made a prototype implementation of our algebra, the Pareto Calculator of Figure 6, with a linear list data structure and using the Simple Cull algorithm for minimisation. Efficient run-time implementations and complexity issues need further study.

10 Conclusions

In this paper, we have introduced an algebra of Pareto points. The algebra supports and describes the incremental, compositional computation of a set of Pareto-optimal configurations and is based on a rigourous definition of the familiar concepts of Pareto-dominance and Pareto-optimality. We have generalised the quantities to partial orders to support compositionality. We have studied the role of cost functions and shown that the generalisation to partial orders makes it possible to use cost functions to make partial design decisions that still leave room for trade-offs to be exploited later. We have given a set of basic operators of the algebra and studied their properties, in particular their properties with respect to minimality and equivalence, to determine whether they can be used effectively in an incremental and compositional design-space exploration and run-time configuration. We have shown two examples of useful higher level operations that can be built from the basic operators of the algebra. With a case-study based on an MPEG-4 delivery chain, we have illustrated how the algebra can be used in practice for run-time multi-dimensional optimisation of dynamic, wireless mobile systems. Future work includes, among other things, further study of the use of the algebra to support run-time decision making in dynamic QoS frameworks. Efficient implementations

will be indispensable to make this possible on resource-constrained embedded devices. We also want to study the use of the algebra and Pareto style trade-offs in general for off-line design flows and design-space exploration.

Acknowledgements

We would like to thank the members of the Betsy consortium for sharing information relating to the modelling of trade-offs and the MPEG-4 case-study and in particular Martin Sénéclauze for the wireless transmission trade-offs and Carolina Blanch for the MPEG-4 profiles.

References

[1] A. Baykasoglu, S. Owen, and N. Gindy. A taboo search based approach to find the Pareto optimal set in multiple objective optimisation. *Journ. of Engin. Optimization*, 31:731–748, 1999.

[2] J. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, April 1980.

[3] C. Blanch, H. de Groot, and P. v.d. Stok. Deliverable D1c: Inventory of competing video codecs and a selection of codecs. Technical report, IST-004042 project Betsy, February 2005. `http://www.hitech-projects.com/euprojects/betsy/results.htm`.

[4] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, New York, 2001.

[5] M. Ehrgott and X. Gandibleux. An Annotated Bibliography of Multi-objective Combinatorial Optimization. Technical Report 62/2000, Fachbereich Mathematik, Universität Kaiserslautern, Kaiserslautern, Germany, 2000.

[6] R. Finkel and J. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[7] M. Geilen, T. Basten, B. Theelen, and R. Otten. An algebra of Pareto points. In *Proc. Application of Concurrency to System Design, 5th International Conference, ACSD 2005*, pages 88–97, Los Alamitos, CA, USA, 2005. IEEE Computer Society Press.

[8] T. Givargis, F. Vahid, and J. Henkel. System-level exploration for Pareto-optimal configurations in parameterized system-on-a-chip. *IEEE Trans. VLSI Syst.*, 10(4):416–422, August 2002.

[9] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[10] M. Gries. Methods for evaluating and covering the design space during early design development. *Integration, the VLSI Journal*, 38(2):131–183, 2004.

[11] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476, 1975.

[12] C. Mattson, A. Mullur, and A. Messac. Minimal representation of multiobjective design space using a smart Pareto filter. In *Proc. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[13] V. Pareto. *Manuale di Economia Politica*. Piccola Biblioteca Scientifica, Milan, 1906. Translated into English by Ann S. Schwier (1971), Manual of Political Economy, MacMillan, London.

[14] M. Sun and R. E. Steuer. Quad-trees and linear lists for identifying nondominated criterion vectors. *ORSA Journal on Computing*, 8(4):367–375, 1996.

[15] R. Szymanek, F. Catthoor, and K. Kuchcinski. Time-energy design space exploration for multi-layer memory architectures. In *Proc. Design Automation and Test in Europe (DATE) 2004*, pages 318–323. IEEE, 2004.

[16] F. Thoen and F. Catthoor. *Modeling, Verification and Exploration of Task-Level Concurrency in Real-Time Embedded Systems*. Kluwer, 2000.

[17] M. Voorneveld. Characterization of Pareto dominance. *Operations Research Letters*, 31(1):7–11, January 2003.

[18] P. Yang and F. Catthoor. Pareto-optimization-based run-time task scheduling for embedded systems. In *Proc. Int. Conf. On Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2003*, pages 120–125. ACM, 2003.

[19] M. Yukish. *Algorithms to Identify Pareto Points in Multi-Dimensional Data Sets*. PhD thesis, Pennsylvania State University, August 2004.

[20] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, November 1999.

[21] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. G. D. Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. on Evolutionary Computation*, 7(2):117–132, April 2003.