DEFINING SOLUTION SET QUALITY

by

Henrik Esbensen

Memorandum No. UCB/ERL M96/1

17 January 1996

DEFINING SOLUTION SET QUALITY

by

Henrik Esbensen

Memorandum No. UCB/ERL M96/1

17 January 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

Traditional algorithms for multi-objective optimization outputs a single solution representing a specific tradeoff of the cost dimensions considered. In contrast, some recent algorithms explores the design space and outputs a *set* of alternative solutions. Such approaches have a number of advantages over traditional methods. However, a fundamental and still unsolved problem in this context is that of evaluating the relative performance of set-generating algorithms, which requires a method of comparing *sets* of solutions. In this report a generally applicable solution set quality measure is proposed. Properties of the measure are proven and examined empirically.

1 Introduction

The traditional formulation of multi-objective optimization problems arising in VLSI layout generation (and many other combinatorial problems as well) is that of minimizing a single-valued cost function aggregating some cost criteria subject to bounds on other criteria. The cost function is typically a weighted sum. While convenient from an algorithmic point of view, such formulations may be very difficult to apply in practice since it may be hard or even impossible to define suitable weights and bounds [3, 5, 7].

Some recent approaches for various VLSI layout problems address this problem [1, 3, 4, 5, 11]. They explicitly explore the design space and generate a set of alternative solutions representing distinct, good tradeoffs, from which the user can then make a final choice. However, a fundamental but still unsolved problem is how to properly compare the performance of design space exploration approaches [8]. Such comparisons requires a method for comparing *sets* of solutions, all of which represent "good" tradeoffs of the cost dimensions.

In this report a solution set quality measure is proposed which allow set-generating algorithms to be evaluated. The measure is independent of the search algorithm used and therefore applies to the evaluation of any set-generating approach, e.g., a random walk, an exhaustive search, or approaches based on branch-and-bound, simulated annealing or the genetic algorithm (GA). The measure can also be used to measure the progress of algorithms based on iterative improvement of (one or more) solutions, such as e.g. the GA. It is consistent with the preference relations used in [4, 5] to drive a GA-based algorithm for building-block placement. In particular, the users specifications of ideal and infeasible solutions are appropriately incorporated, as will be discussed in the following.

The remaining of this report is organized as follows: Section 2 presents a relation on the cost space allowing individual solutions to be compared. The solution set quality measure is introduced in Section 3 and some properties of the measure are proven. Section 4 discuss the practical application of the measure and an empirical validation using constructed sets is presented in Section 5. Finally, conclusions and directions for future work are given in Section 6.

2 Comparing Single Solutions

Let Π be the finite search space considered and let n be the number of distinct optimization criteria considered. Assume without loss of generality that all criteria are to be minimized. The cost measure is the vector-valued function $c : \Pi \mapsto \Re_+^n$, $c(x) = (c(x)_1, \ldots, c(x)_n)$, where $\Re_+ = [0, \infty[$.

As mentioned in Section 1, the traditional way of combining the $c(x)_i$ values into a single-valued cost measure using weights and/or bounds causes some practical problems to be avoided. Therefore, a different approach is taken. An ordering $\prec$ on Π is introduced, which allow individual solutions to be compared without aggregating the cost values. Let $\Re_{+\infty} = [0, \infty]$ and $G = \{(g, f) \in \Re_{+\infty}^n \times \Re_{+\infty}^n | \forall i : g_i \leq f_i\}$. Instead of weights and bounds, the user specifies preferences by defining a *goal and feasibility vector pair* $(g, f) \in G$. For the i'th criterion, g_i is the maximum value wanted, if obtainable, while f_i specifies a limit beyond which solutions are of no interest. For example, if the i'th criterion minimized by an IC placement algorithm is layout area, $g_i = 20$ and $f_i = 100$ states that an area of 20 or less is wanted, if it can be obtained, while

an area larger than 100 is unacceptable. Areas between 20 and 100 are acceptable, although not as good as hoped for. Since the values of (g, f) need *not* be obtainable, in contrast to traditional bounds, they are significantly easier to specify.

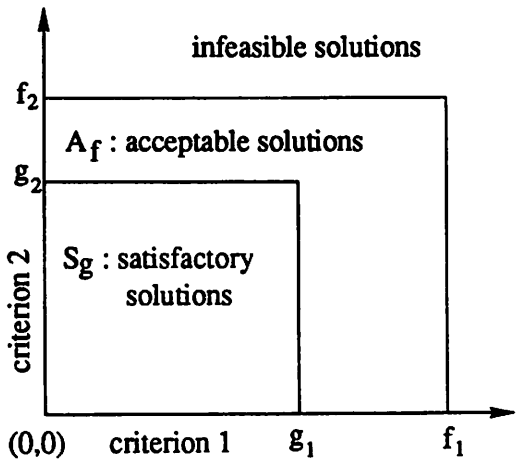

Figure 1: *The sets of satisfactory and acceptable solutions, illustrated in two dimensions.*

For $(g, f) \in G$, let $S_g = \{x \in \Pi \mid \forall\, i : c(x)_i \leq g_i\}$ and $A_f = \{x \in \Pi \mid \forall\, i : c(x)_i \leq f_i\}$ be the set of *satisfactory* and *acceptable* solutions, respectively, as illustrated in Fig. 1. Clearly $S_g \subseteq A_f \subseteq \Pi$ for all $(g, f) \in G$. Finally, the notion of *dominance* is needed. Let $x, y \in \Pi$. The relation x *dominates* y, written $x <_d y$, is defined by

$$x <_d y \quad \Leftrightarrow \quad (\forall\, i : c(x)_i \leq c(y)_i) \wedge (\exists\, i : c(x)_i < c(y)_i) \tag{1}$$

An ordering on Π can now be defined:

Definition 1 (Preference Relation) *Let $x, y \in \Pi$. The relation x is preferable to y with respect to $(g, f) \in G$, written $x \prec_{(g,f)} y$, is defined as follows: If x satisfy all goals, i.e., $x \in S_g$, then*

$$x \prec_{(g,f)} y \quad \Leftrightarrow \quad (x <_d y) \vee (y \notin S_g) \tag{2}$$

If x satisfies none of the goals, i.e., $\forall\, i : c(x)_i > g_i$ then

$$x \prec_{(g,f)} y \quad \Leftrightarrow \quad (x <_d y) \vee [(x \in A_f) \wedge (y \notin A_f)] \tag{3}$$

Finally, x may satisfy some but not all goals.
Assume[1] *that* $(\forall i < k : c(x)_i \leq g_i) \wedge (\forall i \geq k : c(x)_i > g_i)$. *Then*

$$
\begin{aligned}
x \prec_{(g,f)} y \quad \Leftrightarrow \quad & [(\forall i \geq k : c(x)_i \leq c(y)_i) \wedge (\exists i \geq k : c(x)_i < c(y)_i)] && (4)\\
& \vee \\
& [(x \in A_f) \wedge (y \notin A_f)] && (5)\\
& \vee \\
& [(\forall i \geq k : c(x)_i = c(y)_i) \wedge && (6)\\
& \quad \{((\forall i < k : c(x)_i \leq c(y)_i) \wedge (\exists i < k : c(x)_i < c(y)_i)) && (7)\\
& \quad \vee \\
& \quad (\exists i < k : c(y)_i > g_i)\}] && (8)
\end{aligned}
$$

Notice that in (4) only cost values of non-satisfactory dimensions are considered. I.e., when two solutions satisfy the same subset of goals, they are considered equal wrt. the satisfactory dimensions, regardless of their specific cost values in these dimensions. However, in the special case where two solutions are equal wrt. all non-satisfactory dimensions, cf. (6), their cost values in satisfactory dimensions determines their ordering, cf. (7) and (8).

The relation introduced in Definition 1 is an extension of the relation introduced in [9][2], adding the feasibility vector f and the notion of acceptable solutions, A_f. In [4] the relation is applied to control the search of a GA-based algorithm for building-block placement, and the extension of the relation is observed to be of significant practical value. f restricts the GA-based search to the region of practical interest, i.e., prevents the algorithm from wasting time exploring solutions which are non-dominated but in practice infeasible.

When it is clear which goal and feasibility vectors (g, f) are used, $\prec_{(g,f)}$ is written $\prec$. Furthermore, $\neg(x \prec y)$ is written $x \not\prec y$. The extended definition of $\prec$ is transitive, as is intuitively needed:

Property 1 (Transitivity)

$$\forall (g, f) \in G,\ \forall x, y, z \in \Pi : x \prec y \wedge y \prec z \Rightarrow x \prec z \tag{9}$$

Proof : As mentioned above, $\prec$ is a modification of the relation introduced in [9], denoted here by $\prec_p$. Specifically, comparing to [9] it can be seen that by construction

$$\forall x, y \in \Pi : x \prec y \quad \Leftrightarrow \quad (x \prec_p y) \vee (x \in A_f \wedge y \notin A_f) \tag{10}$$

Given $(g, f) \in G$, $x, y, z \in \Pi$, and assume $x \prec y \wedge y \prec z$. Then

$$[(x \prec_p y) \vee (x \in A_f \wedge y \notin A_f)] \quad \wedge \quad [(y \prec_p z) \vee (y \in A_f \wedge z \notin A_f)] \tag{11}$$

For (11) to be true, truth values can be assigned to the clauses of the expression in a total of nine distinct ways. Four of these implies $y \notin A_f \wedge y \in A_f$ and hence are impossible. Three others

[1]Since this can always be obtained by ordering the optimization criteria, full generality is preserved.

[2]A concept of priorities is also introduced in [9]. The relation presented here is an extension of the relation in [9] when all dimensions are given the same priority.

implies $x \prec_p y \wedge y \prec_p z$. Since $\prec_p$ is shown in [9] to be transitive, $x \prec z$ then follows from (10). Two cases remain :

Case a) $x \in A_f \wedge y \notin A_f \wedge y \prec_p z$:

$y \notin A_f \Rightarrow y \notin S_f$. If y satisfies no goals, it follows from the definition of $y \prec_p z$ that $y <_d z$ and hence that $z \notin A_f$, which implies $x \prec z$ due to (10). If y satisfies some but not all goals, $\exists k : (\forall i < k : c(y)_i \leq g_i) \wedge (\forall i \geq k : c(y)_i > g_i)$. Then $y \notin A_f \Rightarrow \exists j \geq k : c(y)_j > f_j$. Either (4) or (6) in the definition of $y \prec_p z$ is true. In either case, $c(y)_j \leq c(z)_j \Rightarrow c(z)_j > f_j \Rightarrow z \notin A_f \Rightarrow x \prec z$.

Case b) $x \prec_p y \wedge y \in A_f \wedge z \notin A_f$:

Substituting x for y and y for z, it was shown in case a) that $x \notin A_f \wedge x \prec_p y \Rightarrow y \notin A_f$. By negation, $y \in A_f \Rightarrow x \in A_f \vee x \not\prec_p y$. Since $y \in A_f \wedge x \prec_p y$ we then have $x \in A_f$, from which $x \prec z$ follows due to (10).

□

Using $\prec$, a given set of solutions can now be ranked: Define the mapping $r : \Pi \times 2^{\Pi} \mapsto N$ by $\forall y \in \Pi, \forall X \subseteq \Pi : r(y,X) = |\{x \in X | x \prec y\}|$. $r(y,X)$ is the rank of y with respect to X. Furthermore, let $X_0 = \{x \in X | r(x,X) = 0\} \subseteq X$. I.e., X_0 is the subset of best solutions in X with respect to $\prec$.

The relation $\prec$ have the following additional properties:

1. $\forall x \in \Pi : x \not\prec x$, i.e., $\prec$ is not reflexive. This follows from (2) through (8).

2. From transitivity : $\forall x, y \in \Pi : x \prec y \wedge y \prec x \Rightarrow x \prec x$. But since $\prec$ is not reflexive, $\forall x, y \in \Pi : \neg(x \prec y \wedge y \prec x)$. Consequently, $\prec$ *is* antisymmetric (since the assumption of antisymmetry is always false), but since $\prec$ is not reflexive, it is not a partial order [2].

3. $\forall x, y \in \Pi : x \prec y \wedge x \notin S_g \Rightarrow y \notin S_g$. This follows from (2) : $y \in S_g \Rightarrow y \prec x$, contradicting $\neg(x \prec y \wedge y \prec x)$ above.

4. $\forall x, y \in \Pi : x \prec y \wedge x \notin A_f \Rightarrow y \notin A_f$. This follows similarly from (2), (3) and (5) : $y \in A_f \Rightarrow y \prec x$, contradicting $\neg(x \prec y \wedge y \prec x)$ above.

5. $\forall X \in \Pi : X \cap S_g \neq \emptyset \Rightarrow X_0 \subseteq S_g$. This follows from (2) : $y \notin S_g \Rightarrow \forall x \in X \cap S_g : x \prec y \Rightarrow r(y,X) > 0 \Rightarrow y \notin X_0$.

6. $\forall X \in \Pi : X \cap A_f \neq \emptyset \Rightarrow X_0 \subseteq A_f$. This follows similarly from (2), (3) and (5) : $y \notin A_f \Rightarrow \forall x \in X \cap A_f : x \prec y \Rightarrow r(y,X) > 0 \Rightarrow y \notin X_0$.

7. In the special case of $g = (0, \ldots, 0)$ and $f = (\infty, \ldots, \infty)$ it can be seen from (2) through (8) that $x \prec y$ is equivalent to $x <_d y$.

8. In the special case of $n = 1$, i.e., one-dimensional optimization, it can be seen from (2) and (3) that $x \prec y$ is equivalent to $x < y$, regardless of $(g, f) \in G$.

3 Comparing Sets of Solutions

From one or more given sets of solutions the user will, sooner or later, select a single specific solution as the "best". For example, in the case of circuit design, a single layout will ultimately be selected for production. However, it is not known *how* the user makes the final choice; it depends on preferences which may never be explicitly expressed. But independently of the final selection method, if the "best" solution belongs to set X rather than set Y, then set X was more valuable to the user than Y and hence should intuitively be better than Y.

The basic idea of the proposed solution set quality measure is to model the final selection performed by the user by an explicit selection function which is parameterized to account for a wide range of possible preferences wrt. the relative importance of the optimization criteria considered. By systematically varying the parameters of the selection function, a class of functions corresponding to a wide range of possible user-preferences is obtained, and the quality of solution set X is then defined as the expected value of the selection function when selecting from X while traversing the class of selection functions.

Assume that the user who ultimately selects a single solution also defines (g, f) used by some algorithm to generate the set(s) from which the final selection is made. The model of the selection process should then be consistent with the preference relation $\prec_{(g,f)}$, assuming that the users actions are consistent. Obtaining this consistency will guide the construction of the set quality measure.

Before proceeding with a model of the selection process, a normalization procedure is needed. Different optimization criteria will have distinct units, e.g., mm^2, ns, % and \$, and will have absolute values in very different ranges. A normalization is therefore performed initially in order to make comparisons in the multi-dimensional cost space without a priori introducing any bias towards a specific dimension.

Definition 2 (Normalization) *Given $X \subseteq \Pi$ and $(g, f) \in G$. Let $c_{i,max} = \max_{x \in X}\{c(x)_i\}$, $c_{i,min} = \min_{x \in X}\{c(x)_i\}$ $i = 1, \ldots, n$, and assume[3] that $\forall\, i : c_{i,max} > c_{i,min}$. Furthermore, let $\Omega(X) = \{x \in \Pi | \forall\, i : c_{i,min} \leq c(x)_i \leq c_{i,max}\} \subseteq \Pi$ and let $\epsilon > 0$ be a small, arbitrary constant. Then $\eta : \Re^n_{+\infty} \mapsto [0;1]^n$ is defined by*

$$\forall\ i = 1, \ldots, n \quad : \quad \eta(t)_i = \begin{cases} 0 & \text{if } t_i < c_{i,min} \\ (1 - 2\epsilon)\frac{t_i - c_{i,min}}{c_{i,max} - c_{i,min}} + \epsilon & \text{if } c_{i,min} \leq t_i \leq c_{i,max} \\ 1 & \text{if } t_i > c_{i,max} \end{cases}$$

The normalized cost $\bar{c} : \Omega(X) \mapsto [\epsilon, 1 - \epsilon]^n$ with respect to $\Omega(X)$ is defined as $\bar{c}(x) = \eta(c(x))$. Similarly, the normalized goal vector $\bar{g}$ of g with respect to $\Omega(X)$ is defined as $\bar{g} = \eta(g)$.

By using $\bar{c}$ and the corresponding normalized goal and feasibility vectors $\bar{g}$ and $\bar{f} = \eta(f)$, the effects of different scalings of the cost dimensions are eliminated, allowing cost values to be compared while preserving the relative ordering defined by $\prec$.

[3] If $c_{i,max} = c_{i,min}$ for some i, all solutions in X are equal wrt. the i'th dimension, which can then be eliminated from the comparison. I.e., full generality is preserved.

The final selection performed by the user is modeled by the selection function s_w defined in Definition 3 below. The idea is that s_w should be as simple as possible, while still being consistent with $\prec$ in the sense stated in Property 2. s_w is essentially a weighted sum of all optimization criteria, which is probably the simplest, meaningful selection function. The scaling terms $w \cdot \bar{g}$ and $\sum_{i=1}^{n} w_i$ as well as the max-terms are minimal alterations of a weighted sum required to obtain the consistency with $\prec$. As mentioned earlier, since the user defines both $(g, f) \in G$ and hence $\prec_{(g,f)}$, it seems intuitively reasonable that the final selection performed using s_w should satisfy Property 2.

Definition 3 (Selection Function) *Given $X \subseteq \Pi$, $(g, f) \in G$, and a weight vector $w \in \Re_+^n$. The selection function $s_w : \Omega(X) \mapsto \Re_+$ is defined by*

$$s_w(x) = \begin{cases} \sum_{i=1}^{n} w_i \bar{c}(x)_i & \text{if } x \in S_g \\ w \cdot \bar{g} + \sum_{i=1}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) & \text{if } x \in A_f \setminus S_g \\ w \cdot \bar{g} + \sum_{i=1}^{n} w_i + \sum_{i=1}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) & \text{if } x \notin A_f \end{cases}$$

where $\bar{c}$ and $\bar{g}$ are normalized with respect to $\Omega(X)$ according to Definition 2.

Property 2 (Selection Function Consistency) *Given $s_w : \Omega(X) \mapsto \Re_+$ according to Definition 3.*

$$\forall\, x, y \in \Omega(X),\ \forall\, w \in \Re_+^n : x \prec y \Rightarrow s_w(x) \leq s_w(y)$$

Proof : First observe from Definition 2 that each coordinate function of η is nondecreasing, i.e.,

$$\forall\, i = 1, \ldots, n,\ \forall\, u, v \in \Re_{+\infty}^n : u_i \leq v_i \quad \Rightarrow \quad \eta(u)_i \leq \eta(v)_i \tag{12}$$

Consequently, $\forall\, x \in S_g \cap \Omega(X)$, $\forall\, w \in \Re_+^n : c(x)_i \leq g_i \Rightarrow \bar{c}(x)_i \leq \bar{g}_i \Rightarrow \sum_{i=1}^{n} w_i \bar{c}(x)_i \leq w \cdot \bar{g}$. Since all terms in the definition of s_w are positive,

$$\forall\, x \in S_g \cap \Omega(X),\ \forall\, y \notin S_g \cap \Omega(X),\ \forall\, w \in \Re_+^n : s_w(x) \leq s_w(y) \tag{13}$$

Similarly, $\forall\, x \in \Omega(X) : \max(\bar{c}(x)_i, \bar{g}_i) \leq 1 \Rightarrow \sum_{i=1}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) \leq \sum_{i=1}^{n} w_i$, and therefore, from the definition of s_w,

$$\forall\, x \in A_f \cap \Omega(X),\ \forall\, y \notin A_f \cap \Omega(X),\ \forall\, w \in \Re_+^n : s_w(x) \leq s_w(y) \tag{14}$$

Given fixed $x, y \in \Omega(X)$, $w \in \Re_+^n$ and assume $x \prec y$. First consider the case $x \in S_g$. If $y \notin S_g$, then $s_w(x) \leq s_w(y)$ follows from (13). If $y \in S_g$, then from (2) and (12), $x <_d y \Rightarrow \forall\, i : c(x)_i \leq c(y)_i \Rightarrow s_w(x) = \sum_{i=1}^{n} w_i \bar{c}(x)_i \leq \sum_{i=1}^{n} w_i \bar{c}(y)_i = s_w(y)$. Then consider the case $x \notin S_g$. If $x \in A_f \setminus S_g$ then $y \notin S_g$ according to property no. 3 on p. 5. On the other hand, if $y \notin A_f$ the result follows from (14). Similarly, if $x \notin A_f$, then $y \notin A_f$ according to property no. 4 on p. 5. Hence, only two possibilities remain:

$$(x \in A_f \setminus S_g \wedge y \in A_f \setminus S_g) \ \vee \ (x \notin A_f \wedge y \notin A_f) \tag{15}$$

In either case of (15), the same sub-expression of s_w is used when computing $s_w(x)$ and $s_w(y)$. Hence, it follows from Definition 3 that it is sufficient to show that

$$\sum_{i=1}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) \;\leq\; \sum_{i=1}^{n} w_i \max(\bar{c}(y)_i, \bar{g}_i) \tag{16}$$

for all x and y satisfying (15). Assuming (15), consider the case when x does not satisfy any goal. Then from (3), $x <_d y \Rightarrow \forall i : \bar{c}(x)_i \leq \bar{c}(y)_i$, from which (16) follows. Finally, assume that x satisfies some but not all goals. As in Definition 1, assume without loss of generality that $(\forall i < k : c(x)_i \leq g_i) \wedge (\forall i \geq k : c(x)_i > g_i)$ for some k, $2 \leq k \leq n$. Since either (4) or (6) holds, we have

$$\begin{aligned}
\sum_{i=1}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) &= \sum_{i=1}^{k-1} w_i \max(\bar{c}(x)_i, \bar{g}_i) + \sum_{i=k}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) \\
&\leq \sum_{i=1}^{k-1} w_i \bar{g}_i + \sum_{i=k}^{n} w_i \max(\bar{c}(x)_i, \bar{g}_i) \\
&\leq \sum_{i=1}^{k-1} w_i \max(\bar{c}(y)_i, \bar{g}_i) + \sum_{i=k}^{n} w_i \max(\bar{c}(y)_i, \bar{g}_i) \\
&= \sum_{i=1}^{n} w_i \max(\bar{c}(y)_i, \bar{g}_i)
\end{aligned}$$

□

Using s_w a solution quality measure $q(Y)$ of a set Y can now be defined. The idea is to vary the weight vector w used in s_w over a user-defined space W, and then define $q(Y)$ as the expected value of s_w over W when selecting solutions from Y. In other words, the quality of a set is the expected selection function value when varying the weights of the selection function corresponding to a wide range of possible preferences of the user. Notice that a smaller value of the quality measure q means a higher set quality.

Definition 4 (Set Quality Measure) *Given $s_w : \Omega(X) \mapsto \Re_+$ according to Definition 3 and a weight space $W \subseteq \Re_+^n$. The set quality measure $q : 2^{\Omega(X)} \mapsto \Re_+$ is defined by*

$$\forall\; Y \subseteq \Omega(X) \;:\; q(Y) = E(\min_{y \in Y}\{s_w(y)\})$$

where the expected value E is computed over all $w \in W$.

As an immediate consequence of Property 2, $q(Y) = q(Y_0)$, i.e., only solutions in Y_0 contributes to $q(Y)$, which is intuitively reasonable. Furthermore, the definition of q is independent of the weight space W. Hence, W can be discrete or continuous, and the weights can be independent or dependent. For example, W can be defined by assuming that w_i is normally distributed $N(\mu_i, \sigma_i)$, or uniform on $[0, m_i]$. Any available knowledge of the preferences used when making the final choice of a solution can thus be utilized when defining W. Due to the normalization performed according to Definition 2, a reasonable weight space can be defined without having

any knowledge of the absolute cost values of each dimension. For example, allowing each w_i to take on any value in $[0,1]$ is sufficient to capture all relative priorities of the criteria.

Properties 3 and 4 below states two properties of the solution set quality measure, which are intuitively required. Property 3 states that by adding a solution to a set, the quality will be unchanged or improve. Property 4 states that when set Y is preferable to set Z in the obvious sense that the best solutions of $Y \cup Z$ all belongs to Y, then Y is at least as good as Z.

Property 3 *Given $q : 2^{\Omega(X)} \mapsto \Re_+$ according to Definition 4.*

$$\forall Y \subseteq \Omega(X), \forall y \in \Omega(X) : q(Y \cup \{y\}) \leq q(Y)$$

Proof : Let $q : 2^{\Omega(X)} \mapsto \Re_+$, $Y \subseteq \Omega(X)$ and $y \in \Omega(X)$ be given. Since the minimum value of any set is non-increasing as the set is expanded,

$$\forall w \in \Re_+^n : \min_{x \in Y \cup \{y\}} \{s_w(x)\} \leq \min_{x \in Y} \{s_w(x)\} \tag{17}$$

$q(Y \cup \{y\}) \leq q(Y)$ then follows from the properties of the expected value E, independently of the weight space W.

□

Property 4 *Given $q : 2^{\Omega(X)} \mapsto \Re_+$ according to Definition 4.*

$$\forall Y, Z \subseteq \Omega(X) : (Y \cup Z)_0 = Y_0 \Rightarrow q(Y) \leq q(Z)$$

Proof : Let $q : 2^{\Omega(X)} \mapsto \Re_+$ and $Y, Z \subseteq \Omega(X)$ be given and assume $(Y \cup Z)_0 = Y_0$. Similar to the proof of Property 3, it is sufficient to show that

$$\forall w \in \Re_+^n : \min_{y \in Y} \{s_w(y)\} \leq \min_{z \in Z} \{s_w(z)\} \tag{18}$$

For a fixed $w \in W$, let $z' \in Z$ be an element minimizing s_w, i.e., $s_w(z') \leq \min_{z \in Z}\{s_w(z)\}$. Since no solution in $Y \cap Z$ can cause a violation of (18), assume without loss of generality that $Y \cap Z = \emptyset$. Then $(Y \cup Z)_0 = Y_0 \Rightarrow \forall z \in Z$, $\exists y \in Y_0 : y \prec z$, and consequently, $\exists y' \in Y_0 : y' \prec z'$. From Property 2 we have $s_w(y') \leq s_w(z')$ from which (18) follows since $\min_{y \in Y}\{s_w(y)\} \leq s_w(y')$.

□

Intuitively it seems desirable that adding a solution y to X should be guaranteed to improve the set quality if $r(y, X) = 0$. Similarly, when set Y contains solutions dominating every solution in set Z, set Y should be strictly better than Z. In other words, the inequalities in Properties 3 and 4 should ideally be strict, rather than allowing equality. This limitation is a consequence of the general shortcoming of the proposed set quality measure that non-convex points of the cost tradeoff surface are not credited. More specifically, given y and X such that $r(y, X) = 0$, if $c(y)$ is non-convex relative to the cost of the solutions X_0, then $q(X \cup \{y\}) = q(X)$. The reason is that s_w is essentially a weighted sum, and the minimum of a weighted sum can never correspond to

a non-convex point, regardless of the weights [7]. Therefore, y never minimizes s_w. The problem of crediting non-convex points is further discussed in Sections 5 and 6.

This Section is concluded by a note on the relationship between the set quality measure presented here and the measure introduced in previous work [10]. The basic idea of the set quality measure in [10] is to combine four intuitively reasonable quality indicators. The four indicators are a) the distance from the center of gravity of the set to the origin, b) the average distance between points having zero rank (diversity), c) the number of solutions having zero rank and d) the volume of a bounding box of the rank zero solutions. These four indicators were combined into a single-valued measure in a simple fashion, using multiplication and division only. However, while each of the indicators seem intuitively reasonable, the combined measure did not properly reflect our subjective notion of solution set quality [10]. The main problem is to device a proper combination of the indicators.

The measure proposed in this report differs fundamentally from the one in [10]. While the latter explicitly states desirable properties of a set, the measure proposed here is implicit in the sense that no explicit definitions of quality indicators are made. However, minimizing set quality as defined in Definition 4 also promotes the optimization of the four individual indicators introduced in [10], and in this sense the measures are consistent. Overall, the measure proposed here is believed to be better than that of [10] since it closer reflects our intuitive notion of set quality. Properties 3 and 4 are essential in this context.

4 Practical Issues of Set Comparison

Assume that two stochastic algorithms A and B have been executed m times each, generating solution sets $A_1, \ldots, A_m$ and $B_1, \ldots, B_m$, respectively. The practical computation of $q(A_1), \ldots, q(A_m), q(B_1), \ldots, q(B_m)$ relative to a goal and feasibility vector pair $(g, f) \in G$ is described in this Section. When all set quality values are known, questions such as "Is A better than B ?" can be answered by applying statistics on the set quality values. A and B can be based on any search strategy, and need not apply the same strategy. If the use of goal and feasibility values are considered inappropriate for the comparison, $g = (0, \ldots, 0)$ and $f = (\infty, \ldots, \infty)$ can be applied, which in effect eliminates the notions of goals and feasibility.

The solution quality values are determined by going through the following steps:

1. Normalize the cost dimensions according to Definition 2, using

$$X = \bigcup_{i=1}^{m} (A_i \cup B_i)$$

   and any small ϵ, e.g. $\epsilon = 10^{-4}$. Also normalize the goal vector. By construction, $\Omega(X)$ is now large enough for all remaining computations to be well-defined. The quality $q(V)$ of a new set V can be computed using the same normalization as long as $V \subseteq \Omega(X)$. However, if $V \not\subseteq \Omega(X)$, it is necessary to re-normalize all sets considered.

2. Define a weight space W. There are no restrictions on the definition of W, any available knowledge of the selection process can be incorporated, cf. Section 3.

3. Depending on the choice of W, an expression for the exact value of q may be obtainable and can then be used to accurately calculate q for each solution set. However, a simpler approach which applies to any definition of W is to compute an estimate $\hat{q}$ of q by sampling N points. I.e., for each set Y, estimate $q(Y)$ by

$$\hat{q}(Y) = \frac{1}{N} \sum_{k=1}^{N} \min_{y \in Y} \{s_{w(k)}(y)\}$$

   where each vector $w(k)$ is generated independently according to the chosen definition of W. Assuming a sufficiently large sample size N, this estimation method is statistically sound. Furthermore, Properties 3 and 4 still holds when substituting q by $\hat{q}$, as a consequence of (17) and (18), respectively.

5 Experimental Results

In this Section the proposed quality measure is empirically validated using constructed sets. The measure is used in [6] to compare the performance of a GA for building-block placement with a random walk.

Assume first that $n = 2$, $g = (0,0)$ and $f = (\infty, \infty)$. Eight constructed sets s1 through s8 are plotted in Fig. 2. The quality of each set is estimated as described in Section 4, using $W = [0,1]^2$, assuming that the weights are independent and uniform on $[0,1]$. A sample size of $N = 10,000$ is used when computing $\hat{q}$.

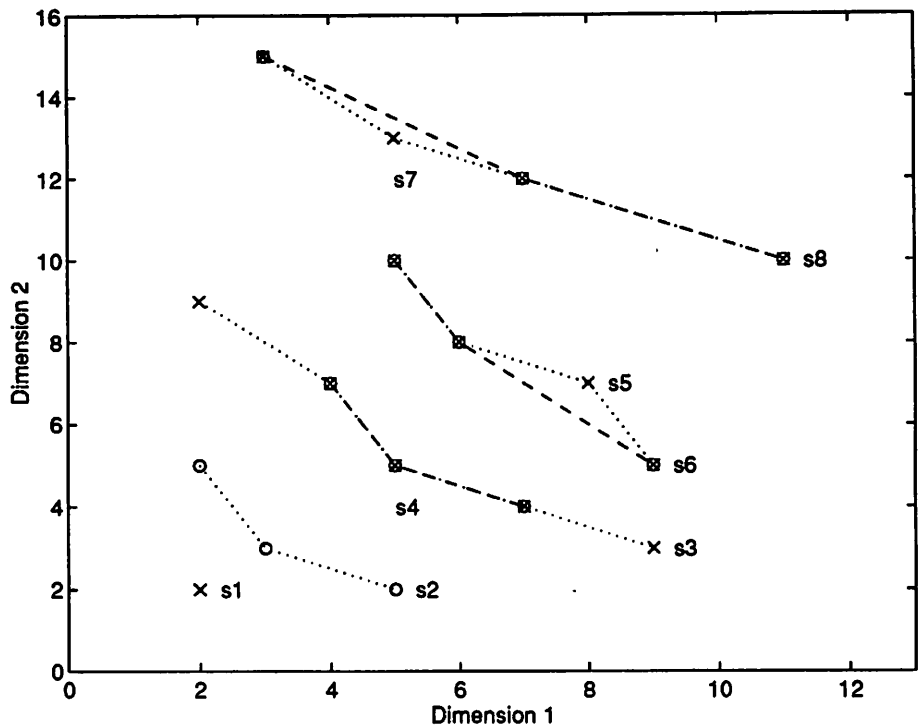

Figure 2: *Constructed sets s1 through s8. Each set is indicated by either circles connected by a dotted line or crosses connected by a dashed line. s4 is a subset of s3, s6 a subset of s5 and s8 a subset of s7.*

Set	s1	s2	s3	s4	s5	s6	s7	s8
Size	1	3	5	3	4	3	4	3
$\hat{q}$	0.0001	0.0768	0.2143	0.2655	0.4149	0.4149	0.5270	0.5291

Table 1: *Estimated solution set qualities corresponding to Fig. 2.*

When computing $\hat{q}$ a solution may never be sampled, i.e. never minimize s_w. This will happen if either 1) the solution is non-convex, cf. Section 3, or 2) the weight w needed for the solution to minimize s_w was never generated during the computation of $\hat{q}$. In the case of sets s1 through s8, all non-sampled points are non-convex.

As can be seen from Table 1, except for sets s5 and s6, the set quality decreases strictly with the set number, which seems intuitively reasonable. However, set s5 is not better than s6 because the solution with cost (8,7) is non-convex relative to s5. Solutions (7,4) and (4,7) of set s3 and (7,12) of s7 are similarly not sampled due to their non-convexity.

The effect of changing (g, f) to $g = (9, 6)$ and $f = (13, 16)$ is illustrated in Fig. 3 and Table 2, using four new constructed sets s1 through s4. Using these values of (g, f) has the effect of changing the relative quality of the sets from the order s2, s1, s4, s3 (decreasing quality) to the order s1, s2, s3, s4, as is intuitively desirable.

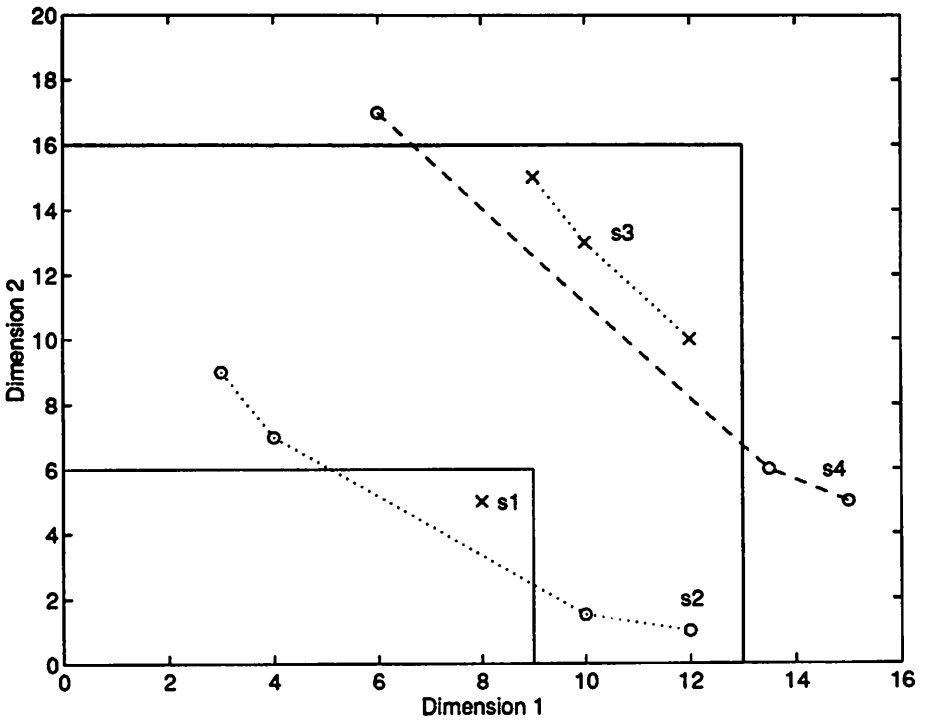

Figure 3: *Constructed sets s1 through s4. The boxes indicates S_g and A_f using $g = (9, 6)$ and $f = (13, 16)$.*

Set	s1	s2	s3	s4
Size	1	4	3	3
$\hat{q}_1$	0.3347	0.8393	1.0328	1.9739
$\hat{q}_2$	0.3347	0.1807	0.6249	0.4963

Table 2: *Estimated solution set qualities corresponding to Fig. 3. $\hat{q}_1$ is the estimated qualities using $g = (9, 6)$ and $f = (13, 16)$. $\hat{q}_2$ is computed using $g = (0, 0)$ and $f = (\infty, \infty)$.*

6 Conclusions and Future Work

A solution set quality measure needed to evaluate the performance of set-generating algorithms has been proposed. The measure is independent of the search method used and therefore generally applicable. At the same time, the measure is consistent with a preference relation used to guide GA-based searches, and appropriately incorporates the specification of goal and feasibility vectors. The measure also overcomes some shortcomings of a previously proposed measure. The practical use of the measure has been illustrated on constructed sets and to a large extent the measure coincides with our intuitive notion of set quality. However, non-convex solutions are not appropriately accounted for, which is an important topic of future work. Other issues requiring further investigation includes sampling accuracy and the effect of increasing the number of optimization dimensions.

Acknowledgments

The author would like to thank Dr. Michael A. Lee at the Computer Science Division, Jim Prieger at Dept. of Economics and Imola K. Fodor at Dept. of Statistics, all at University of California, Berkeley, for several useful discussions and suggestions concerning this work. The research was supported by SRC grant no. 95-DC-324, NSF grant no. MIP 91-17328 and the Danish Technical Research Council.

References

[1] J. Cong, Y. Ding, "On Area/Depth Trade-Off in LUT-Based FPGA Technology Mapping," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 2, No. 2, June 1994.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms,* MIT Press, Cambridge, Massachusetts, 1990.

[3] P. Dasgupta, P. Mitra, P. P. Chakrabarti, S. C. DeSarkar, "Multiobjective Search in VLSI Design," *Proc. of The 7th International Conference on VLSI Design*, pp. 395-400, 1994.

[4] H. Esbensen, E. S. Kuh, "Design Space Exploration for Building-Block Placements," Memorandum No. UCB/ERL M95/84, College of Engineering, University of California, Berkeley, CA 94720, Oct. 1995.

[5] H. Esbensen, E. S. Kuh, "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration," *Proc. of the IEEE Multi-Chip Module Conference,* 1996 (to appear).

[6] H. Esbensen, E. S. Kuh, "Design Space Exploration Using the Genetic Algorithm," *Proc. of the IEEE International Symposium on Circuits and Systems,* 1996 (to appear).

[7] P. J. Fleming, A. P. Pashkevich, "Computer Aided Control System Design Using a Multiobjective Optimization Approach," *Proc. of the IEE Control '85 Conference,* pp. 174-179, 1985.

[8] C. M. Fonseca, P. J. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation,* Vol. 3, No. 1, pp. 1-16, 1995.

[9] C. M. Fonseca, P. J. Fleming, "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I : A Unified Formulation," *Research Report 564*, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield S1 4DU, U.K., Jan. 1995.

[10] M. A. Lee, H. Esbensen, L. Lemaitre, "The Design of Hybrid Fuzzy/Evolutionary Multiobjective Optimization Algorithms," *Proc. of the 1995 IEEE/Nagoya University WWW on Fuzzy Logic and Neural Networks / Evolutionary Computation,* pp. 118-125, 1995.

[11] J. Lillis, C.-K. Cheng, T.-T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *Proc. of the International Conference on Computer Aided Design,* pp. 138-143, 1995.