Chapter IV Vector Ordinal Optimization

Consider a multi-objective optimization problem with m objective functions $J_1, \ldots, J_m$ over a finite but huge design space Θ. If the user knows the priority among these objective functions, or furthermore can assign appropriate weights to each objective functions, s/he can reformulate this problem as either a sequence of m single objective optimizations or a single objective optimization using the weighted sum of $J_1, \ldots, J_m$ as the objective function. Then the method introduced in Chapter II will suffice to solve this new problem. However, a more difficult case is that the user does not know the priority or the appropriate weights among the objective functions. In this chapter, we focus on this type of problem. The purpose of the optimization here is to find designs such that the objective functions are, in a sense, minimized. The operative concept in multi-criterion optimization problems, of course, is that of Pareto frontier or non-dominated solutions. All the designs in the Pareto frontier are considered Pareto optimal. The concept of Pareto optimum was formulated by Vilfredo Pareto (Pareto 1896). A design is said to be Pareto-optimal if it is not dominated by any other designs (i.e., there exists no other design that is better for at least one objective function value, and equal or superior with respect to the other objective functions). All Pareto-optimal points constitute the so-called Pareto frontier which plays the same role as maximum or minimum in single criterion optimization. As discussed before in Chapter I, exact values of the m objective functions are often computationally infeasible to obtain via simulation (due to the $1/(n)^{1/2}$ limit) and thus it is often hard to obtain the Pareto frontier. Genetic algorithms and evolutionary algorithms are alternatives (Goldberg 1989), which do not guarantee a set of designs in the Pareto frontier but try to find a set of designs hopefully not too far away from the Pareto frontier (Zitzler et al. 2003) and these methods do not consider the difficulty of time consuming simulation-based performance evaluation. Comprehensive surveys in this area can be found in (Coello 2000; Tan et al. 2002). In this chapter, by generalizing ordinal optimization from the scalar case to the vector case, we aim at quantifying how many observed layers (definition follows) are enough to contain the required number of designs in the Pareto frontier with high probability. Both tenets of the scalar OO are kept:

1. the order we introduced converges exponentially as the number of replications increases.
2. we ask for only some good enough designs that are pareto or nearly pareto optimal.

In Section 1, we first include a very brief review of the traditional vector optimization results for completeness. Then we define the concept of Pareto frontier and layer in vector optimization, the good enough set, selected set, universal alignment probability (UAP), and ordered performance curve (OPC) in the vector case. In Section 2, based on abundant experiments, we give the UAP table for a 2-dimensional case, quantifying subset selection sizes for different types of two-objective optimization problems. Following this idea, one can quantify subset selection sizes when there are an arbitrary number of objective functions. In Section 3, we show the exponential convergence rate of observed layers to true ones. In Section 4, we use examples to show how the above numerical results help to reduce the search efforts for true Pareto frontier by at least one order of magnitude. At last, we summarize in Box 4.1 the general steps to apply Vector Ordinal Optimization (VOO).

1 Definitions, terminologies, and concepts for VOO

First, we include here a very brief review of the traditional vector optimization problem of "Pareto-min $\sum_{i=1}^{m} J_i(\theta)$". To locate a point on the Pareto frontier, we usually resort to some form of scalarization of the vector criteria. The most popular method is to consider a weighted sum of the criteria

$$\min \sum_{i=1}^{m} \lambda_i J_i(\theta), \quad \text{with } \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \tag{4.1}$$

where the λ_i'*s* play the role of LaGrange multipliers. Under appropriate convexity conditions, solutions of the scalarized problem over all λ_i can determine all points on the Pareto frontier. There are also other possible methods of scalairzation. For example, consider

$$\min \sum_{i=1}^{m} \lambda_i (J_i(\theta) - J_i^*(\theta))^2 \tag{4.2}$$

where $J_i^*(\theta)$, called aspiration level, is the desired but unrealizable value of the *i*th performance criterion. Or consider,

$$\min\left\{\max J_i(\theta), i=1,...,m\right\}, \tag{4.3}$$

where we assume the true performances of any two designs are distinguishable in any objective function.

Exercise 4.1: Prove that solutions of Problems in Eqs. (4.1)-(4.3) above all result in a point on the Pareto Frontier.

However, our goals in VOO are somewhat different. We are not that interested in locating one point on the Pareto frontier. In this book, such a task is contrary to the basic tenets of OO. Instead, we want to locate a set of points which are "near or close to" the Pareto frontier as explained in the introduction part of this chapter.

Hence we now introduce definitions and notations necessary for VOO. These definitions parallel, in concept, to those defined in Chapter II for single objective ordinal optimization and can be understood similarly.

- Θ the search space for the optimization variables θ.
- J_i the performance criteria (also called objective functions) for the system. In contrast to the scalar case, we have m performance criteria, i=1, . . , m.
- N the number of designs uniformly chosen from Θ. It is understood that for each choice of θ, there corresponds a set of values $J_i(\theta)$, i=1, 2, ... , m.
- $\prec$ the dominance relation between designs. A design θ_1 is said to dominate θ_2, denoted by $\theta_1 \prec \theta_2$, if $J_i(\theta_1) \leq J_i(\theta_2)$, for i=1, 2, ... , m, with at least one inequality being strict. If θ_1 does not dominate θ_2, θ_2 will be called noninferior to θ_1. Furthermore, if neither $\theta_1 \prec \theta_2$ nor $\theta_2 \prec \theta_1$ is true, θ_2 and θ_1 will be called incomparable.
- $\mathcal{L}_1$ Pareto Frontier. A set of designs $\mathcal{L}_1$ is said to be in the Pareto frontier, in terms of the objective functions J_1, ... , J_m, if it contains all the designs that are not dominated by other designs in the design space Θ; i.e.,

$$\mathcal{L}_1 \equiv \{\theta | \theta \in \Theta, \nexists\, \theta' \in \Theta, \text{ s.t. } \theta' \prec \theta\}.$$

  Designs in Pareto frontier are the counterparts of the true optimal design in the scale case.
- Ω an operator that maps a design space to the set of the Pareto frontier with respect to the objective functions as $\mathcal{L}_1 = \Omega(\Theta)$. The concept of Pareto frontier can be extended to a sequence of layers.
- $\mathcal{L}_i$ layers. A series of designs $\mathcal{L}_{i+1} = \Omega(\Theta \backslash \cup_{j=1,...,i} \mathcal{L}_j)$, i=1, 2, ... , are called layers, where $A \backslash B$ denotes a set containing all the designs included in

set A, but not included in set B. Designs in $\mathcal{L}_i$ are called layer i designs. They are successive Pareto frontiers after the previous layers have been removed from consideration. *The significance of layers is that they introduce a natural order in the design space* Θ and there are *no preferences* on the objective functions and *no preferences* on the designs in the same layer.[1]

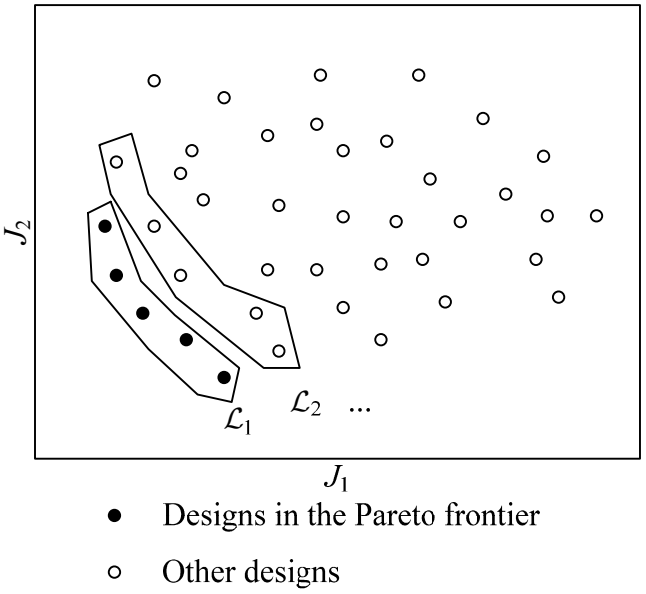


Fig. 4.1. Graphic illustration of layers (assuming minimization)

N_l the number of layers formed by the N designs uniformly chosen from Θ.

$\hat{N}_l$ the number of observed layers formed by the N designs uniformly chosen from Θ. This is a random number and varies in different replications.

$\hat{J}_i$ the observed performance criteria of the sampled designs. With n replications, we denote observed value of i-th performance criterion in

[1]There are other ways to introduce order for multi-objective optimization. For example, it was proposed in (Teng, Lee and Chew 2006) to sort designs according to the number of dominating designs. Pareto Frontier is the set of designs with 0 dominating designs. In that order, although there is no preference on the designs in the first layer (i.e., the Pareto Frontier), there usually are preferences on the designs in the second or other layers.

j-th replication by $\hat{J}_i(\theta, j) = J_i(\theta) + w_{ij}(\theta, \xi)$, j=1,…,n, where w_{ij} are noises. By default, observed performance always refers to the average over all replications:

$$\overline{\hat{J}}_i(\theta) = \frac{1}{n}\sum_{j=1}^{n} \hat{J}_i(\theta, j), i = 1, 2, ..., m .$$

Note, $\overline{\hat{J}}_i(\theta)$ is a random variable whose distribution also depends on the number of replications n.

$\hat{\prec}$ dominance in observation. A design θ_1 is said to dominate θ_2 in observation, denoted by $\theta_1 \hat{\prec} \theta_2$, if $\overline{\hat{J}}_i(\theta_1) \le \overline{\hat{J}}_i(\theta_2)$, i=1, 2, . . . , m, with at least one inequality being strict.

$\hat{\mathcal{L}}_i$ observed layers. Dominance in observation is a stochastic relationship among designs, and will lead to stochastic partition of the design space into the observed layers $\hat{\mathcal{L}}_i$, i=1, 2,…

G good enough set. Defined as the union of the designs in the true first g layers (e.g., when g=1,G is the Pareto frontier $\mathcal{L}_1$). As in scalar OO, the user is free to decide how many layers constitute G.

S selected set. Defined as the designs chosen based on observed performances.

Selection Rule

Only a selection rule similar to the Horse Race rule is considered here. That is to select all designs in the observed first s layers.

$G \cap S$

the set of truly good enough designs in S.

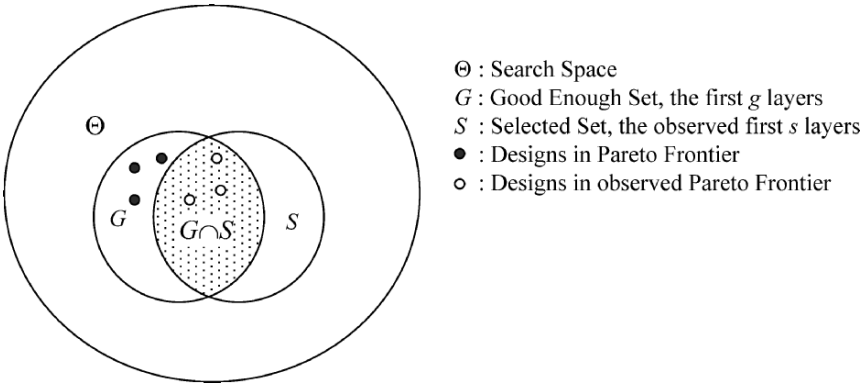


Fig. 4.2. Graphical illustration of Θ, G, and S in VOO

Alignment Probability (AP) $\equiv$Prob[$|G \cap S| \geq k$]

The probability that there are actually k truly good enough designs in S. This is the same as in scalar OO. Based on the notion of layers, it can be written as

$$AP = \text{Prob}\left[\left|\bigcup_{i=1}^{s} \hat{\mathcal{L}}_i \cap \bigcup_{i=1}^{g} \mathcal{L}_i\right| \geq k\right].$$

VOPC

Ordered Performance Curve in the vector case (VOPC). Similar to scalar OO, AP in VOO is also affected by problem types. We introduce VOPC for multi-objective optimization problems. VOPC is described by a function $F(x)$, where x is the layer index, from 1 to the total number of layers of that problem, and $F(x)$ is the number of designs in the first x layers. Correspondingly, we can also focus on the map $f(x)$, which sends the layer index x, ranging from 1 to the total number of layers of that problem, to $f(x)$, the number of designs in the x-th layer. In Fig. 4.3, we use two-objective optimization as an example to show how $f(x)$ describes different types of multi-objective optimization problems. There are three types of $F(x)$ in Fig. 4.3. Each column shows one type of two-objective optimization problem and the corresponding $f(x)$. The true performances of the designs are denoted by dots. Suppose that we uniformly pick up designs to compose the selected set S. In the first type, there are few designs in the Pareto frontier. Then, it is hard for S to contain some designs in the Pareto frontier. This type of optimization problems is hard. The problems in the second and third columns are neutral and easy, respectively. VOPC is a concept to classify the problem type, which is logically similar to OPC classifying the problem type in scalar OO. However, since we do not know the appropriate weight among the multiple objective functions, we cannot use the value of the objective functions to measure the "performance" of the designs in the same layer. Instead, we use the total number of designs in the previous layers as such a measure. Other definitions are possible. We still call it VOPC though we know the "performance" here is neither the value of any objective function nor the value of a weighted sum of these objective functions.

w_i noise/error level in objective function J_i. We assume $w_{ij}(\theta, \xi)$, $j = 1,2,\ldots,n$, form an i.i.d. sequence of random variables with zero mean. When there is no confusion, we simply use w to represent the noise levels.

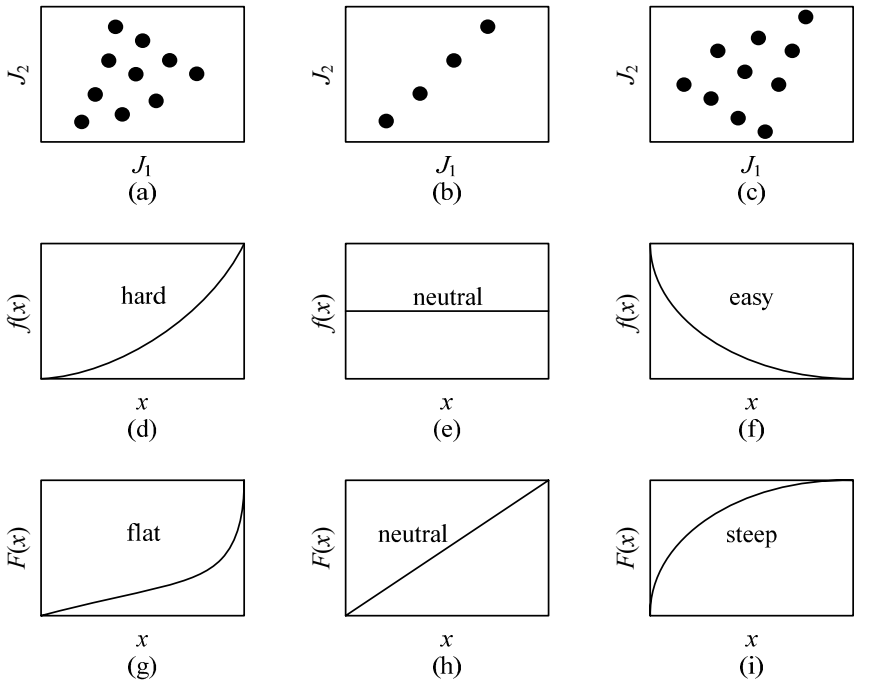


Fig. 4.3. Three types of two-objective optimization problems. A fourth type, the general type, is not shown

The Universal Alignment Probability (UAP)

$$\equiv \text{Prob}\left[\left|G \cap S\right| \geq k \,/ N, N_l, w, \text{VOPC type}\right]$$

$$\equiv \text{UAP}\left(N, N_l, w, \text{VOPC type}\right)$$

As in scalar OO, the alignment probability can be tabulated once N, the number of designs, N_l, the number of layers, the noise/error level and VOPC type of a problem is given. The UAP table for the 2-dimension case will be given in Section 2.

Exercise 4.2: Please compare the concepts of order, good enough set, selected set, ordered performance curve, and universal alignment probability in ordinal optimization when there are one or multiple objective functions.

2 Universal alignment probability

In VOO, we care about the probability that the observed first s layers contain at least k designs of the true first g layers; we want this probability to be greater than or equal to some required confidence probability α, i.e.,

$$\text{Prob}\left[\left|\bigcup_{i=1}^{s}\hat{\mathcal{L}}_i \cap \bigcup_{i=1}^{g}\mathcal{L}_i\right| \geq k\right] \geq \alpha .$$

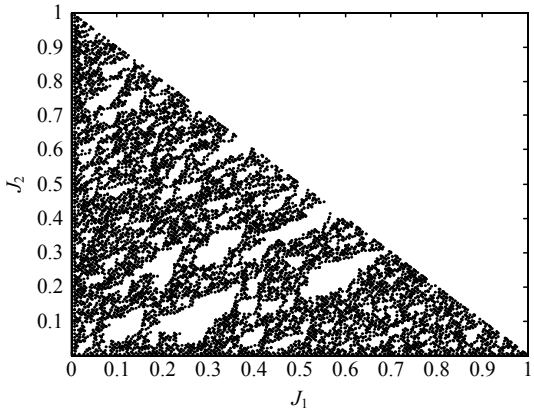
As in the scalar case, *g*, *s*, *k*, VOPC type and the noise level all affect AP. In general, it is difficult to get a closed-form formula to calculate AP, giving the values of these factors. In the scalar case, a table is used to quantify the relationship among *g*, *s*, *k* under different OPC and noise levels (Lau and Ho 1997). In VOO, we similarly tabulate the relationship among *g*, *s*, and *k*. In the rest of this section, we show how to do experiments on two-objective optimization problems as an example. For cases with more than two objective functions, the method is similar. The importance of the two-dimensional UAP table also lies in that, under mild assumption, the VOO-UAP table for two objective functions supplies an upper bound for the size of the selected set when there are more objective functions.

Exercise 4.3: Suppose all the designs are distinguishable in each objective function, i.e., $\forall \theta, \theta' \in \Theta$, i=1,2,…m, $J_i(\theta) \neq J_i(\theta')$. Please show that the Pareto frontier with respect to m-1 objective functions is a subset of the Pareto frontier w.r.t. m objective functions.

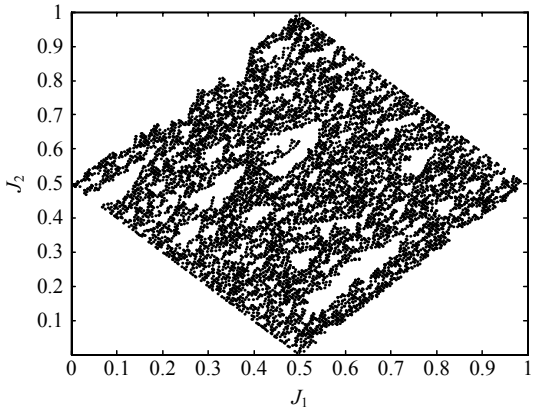
Exercise 4.4: Based on the results in the last exercise, please show that the UAP table for two-objective optimization supplies an upper bound for the size of the selected set when there are more objective functions.

We consider three types of VOPCs in the experiments: Flat, neutral, and steep. Without loss of generality, we assume that the true performance of each design is within [0,1], that there are totally 10000 designs and 100 layers.[2] The numbers of designs in each layer are also specified:

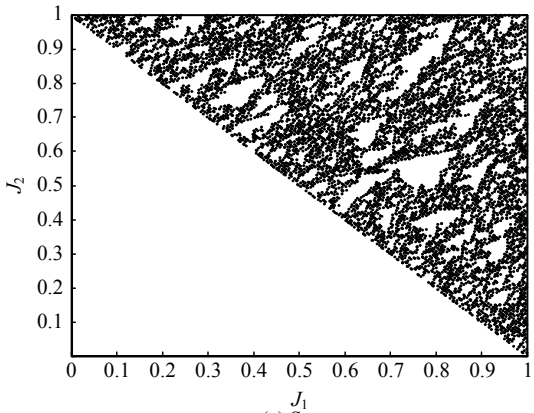
[2]With no prior knowledge on the problem, for the neutral VOPC, we want to ensure the performance vectors of designs are uniformly deployed in an m-dimensional "cubic". So, fixing N_l =100, for m=2, if there are 100 designs in each layer, there will be N=100×100 designs, where N_l is the total number of layers. For the flat and steep VOPC, we want to ensure the performance vectors of the designs are uniformly deployed in an m-dimensional "pyramid". So, fixing N_l =100, for m=2, let there be only one design in the first layer for the flat VOPC (or in the last layer for the steep VOPC), and the numbers of the designs in the successive layers increase (decrease in the steep VOPC) by a constant. For general m, to avoid the curse of dimensionality, fixing N_l =100, we generate N=C(1+ N_l) N_l /2 designs, and C is a positive number depending on m. See Fig. 4.4.



(a) Flat



(b) Neutral



(c) Steep

Fig. 4.4. One example of the randomly generated true performances of the designs for the three types of VOPC in the experiment

- for flat VOPC, $|\mathcal{L}_i|=2i-1$;
- for neutral VOPC, $|\mathcal{L}_i|=100$;
- for steep VOPC, $|\mathcal{L}_i|=201-2i$, $i=1, 2,\ldots, 100$.

We randomly generate the true performance of each design such that the number of designs in each layer meets the above requirements. We show one example in Fig. 4.4. i.i.d. uniformly distributed noises are considered, i.e.,

$$w_{ij}(\theta, \xi) \sim U[-w,w],\ i=1, \ldots, m, j=1,2,\ldots,n.$$

Three noise levels are considered: $w = 0.5, 1.0, 2.5$. The three noise levels are supposed to represent "small, medium, and large". The reason is similar to the ones stated in Section II.5, i.e., consider the neutral type for example, when w=0.5, the worst design barely has the chance to be observed better than the best design; this probability is positive when w=1.0, and much greater when w=2.5. By adding observation noises to the true performances of each design, we can find the observed first s layers. For each type of VOPC, we repeat the above procedure 1000 times to estimate the alignment probability. The values of g, s, k are also specified for each VOPC so that the number of good enough designs does not exceed 20% of the size of the entire design space:

- for flat VOPC, g, $s\in[1, 44]$;
- for neutral VOPC, g, $s\in[1, 20]$;
- for steep VOPC, g, $s\in[1, 10]$;

and $k \in[1, 100]$ for each type. When the alignment probability $\alpha\geq0.95$, we try to describe the value of s as a function of k and g. We find that the following functional form fits well in all cases:

$$Z(k,g) = e^{Z_1} k^{Z_2} g^{Z_3} + Z_4, \tag{4.4}$$

where Z_1, Z_2, Z_3, Z_4 are constants depending on the VOPC types and noise characteristics. We show one example in Fig. 4.5, where the solid lines represent the number of the observed layers to select, which is obtained through the experiments, and the dashed lines represent the prediction given by Eq. (4.4). As we can see, the two lines are close to each other. We perform a regression on the data of (g, s, k) of the experiments, which lead to $\alpha\geq0.95$, and this in turn produces the coefficients appearing in Eq. (4.4). We list the regressed values in Table 4.1.

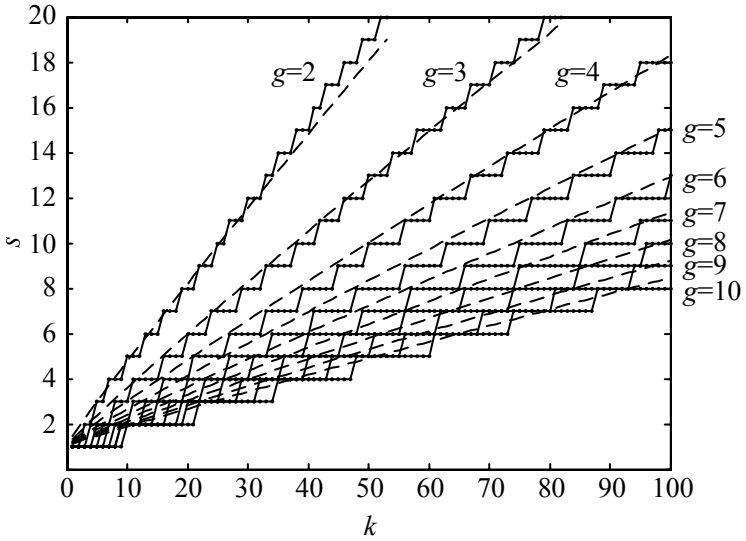


Fig. 4.5. The number of the observed layers to select to ensure AP≥0.95 in the Neutral VOPC and w=0.5. The solid lines represent the true value. The dashed lines represent the predicted value

Table 4.1. Regressed values of Z_1, Z_2, Z_3, Z_4 in $Z(k,g)$ for UAP of VOO

Noise	$U[-0.5, 0.5]$		
OPC class	Flat	Neutral	Steep
Z_1	4.2004	–0.2176	–0.7564
Z_2	1.1953	0.9430	0.9156
Z_3	–2.3590	–0.9208	–0.8748
Z_4	3.1992	1.0479	0.6250
Noise	$U[-1.0, 1.0]$		
OPC class	Flat	Neutral	Steep
Z_1	4.7281	0.3586	–0.1536
Z_2	1.0459	0.8896	0.8721
Z_3	–2.1283	–0.8972	–0.8618
Z_4	2.4815	0.8086	0.5191
Noise	$U[-2.5, 2.5]$		
OPC class	Flat	Neutral	Steep
Z_1	5.2099	0.9379	0.3885
Z_2	0.9220	0.8445	0.8536
Z_3	–1.9542	–0.8890	–0.8847
Z_4	1.9662	0.5946	0.5414

To ensure that Eq. (4.4) and Table 4.1 produces an upper bound estimate of the number of the selected layers, we restrict the numerical ranges as follows:

- for neutral VOPC, $g\in[1, 20]$, $k\in[1, 100]$, $s=Z(\cdot/\cdot)\leq 20$;
- for steep VOPC, $g\in[1, 10]$, $k\in[1, 100]$, $s=Z(\cdot/\cdot)\leq 10$;
- for flat VOPC, when $k \leq 50$, the range of g is $1\leq g \leq 44$. When $50<k \leq 100$, for flat VOPC with noise level $w=0.5$, the range of g is $1\leq g\leq 25$; for noise level $w=1.0$, the range of g is $1\leq g\leq 30$; for noise level $w=2.5$, the range of g is $1\leq g \leq 35$. For flat OPC, all the (g, k) combinations should let $s=Z(\cdot/\cdot)\leq 44$.

Note, in practice, we may have a different sample size N and a different number of layers N_l as we assumed when generating Table 4.1, our idea is to use the total number of observed layers $\hat{N}_l$ as an estimate of N_l and keep the ratios g/N_l, k/N, s/N_l as constant. This idea is demonstrated in Section 4 through examples.

3 Exponential convergence w.r.t order

VOO is based on ordinal comparison as in the scalar OO. That is, the comparison of designs and sorting them into observed layers. As in scalar OO, under mild conditions, it can be shown that ordinal comparison in VOO also has exponential convergence rate, namely the probability that the true i-th layer $\mathcal{L}_i$ is the same as the observed i-th layer $\hat{\mathcal{L}}_i$ is of form $1-O(e^{-n\beta})$, where n is the number of replications and $\beta>0$ is a constant. In fact, when some designs in the i-th layer $\mathcal{L}_i$ is not in the observed i-th layer $\hat{\mathcal{L}}_i$, there must be *at least one pair* of designs, the observed order (dominance) of which is different from the true order. Since the design space is finite, in order to prove the exponential convergence of layers, it is sufficient to show that any pair of designs θ_1 and θ_2 can only change their observed order with an exponentially decaying probability in terms of the number of replications n.

$$\begin{aligned}&\text{Prob}[\mathcal{L}_i=\hat{\mathcal{L}}_i \text{ for all } i]\\ &\geq 1-\text{Prob}[\text{there exist a pair of designs } \theta_1 \text{ and } \theta_2 \text{ changing order in observation}]\\ &\geq 1-\sum_{\theta_1,\theta_2}\text{Prob}\left[\theta_1 \text{ and } \theta_2 \text{ changes order in observation}\right]\end{aligned} \tag{4.5}$$

Once an observation (based on n replications) is made, there are only three possible order relationships between any two designs θ_1 and θ_2:

$\theta_1 \prec \theta_2$, $\theta_1 \succ \theta_2$, and incomparable. For all these three cases, when a change in order happens in observation, there is at least one objective function J_i such that one of the following is true, among m objective functions.

1. $J_i(\theta_1)<J_i(\theta_2)$ and $\overline{\hat{J}}_i(\theta_1) \geq \overline{\hat{J}}_i(\theta_2)$ hold simultaneously.
2. $J_i(\theta_1)>J_i(\theta_2)$ and $\overline{\hat{J}}_i(\theta_1) \leq \overline{\hat{J}}_i(\theta_2)$ hold simultaneously.

In other words, in at least one objective function, a change in order occurs in observation. Thus, we can bound Prob[θ_1 and θ_2 changes order in observation] from above by Prob[$\overline{\hat{J}}_i(\theta_1) \geq \overline{\hat{J}}_i(\theta_2)$] when $J_i(\theta_1)<J_i(\theta_2)$ and Prob[$\overline{\hat{J}}_i(\theta_1) \leq \overline{\hat{J}}_i(\theta_2)$] when $J_i(\theta_1)>J_i(\theta_2)$.

It follows from the exponential convergence w.r.t. order for scalar case in Section II.4.2 (based on Large Deviation Theory) that probability for the order to change in one objective function decreases exponentially as a function of n. In other words, when $J_i(\theta_1)<J_i(\theta_2)$, and as long as the conditions on the samples (or equivalently on the noises) of scalar OO hold, i.e., the moment generating functions $E(e^{sw_{i1}(\theta,\xi)})$ exists for all $s \in (-d,d)$, for some $d>0$, there must be a positive β such that

$$\text{Prob}\left[\overline{\hat{J}}_i(\theta_1) \geq \overline{\hat{J}}_i(\theta_2)\right] = O(e^{-n\beta})$$

and when $J_i(\theta_1)>J_i(\theta_2)$, there must be a positive β such that

$$\text{Prob}\left[\overline{\hat{J}}_i(\theta_1) \leq \overline{\hat{J}}_i(\theta_2)\right] = O(e^{-n\beta}).$$

As a result, we have

$$\text{Prob}\left[\theta_1 \text{ and } \theta_2 \text{ changes order in observation}\right] = O(e^{-n\beta})$$

and furthermore due to Eq. (4.5), we have

$$\text{Prob}[\mathcal{L}_i = \hat{\mathcal{L}}_i \text{ for all } i] = 1 - O(e^{-n\beta}).$$

Exercise 4.5: Another basic idea in single-objective OO is goal softening. What is the advantage to consider goal softening in VOO? Can we show some results similar to Section II.4.3?

4 Examples of search reduction

When we obtain the regressed values in Table 4.1, there are several assumptions:

(A1) There are 10000 designs and 100 layers in total.
(A2) The observation noises of different designs are independent.
(A3) The observation noises have uniform distribution.

It turns out that, even when these assumptions are not met, Table 4.1 still gives a good guidance on how many observed layers should be selected due to its universality. We present two examples here to demonstrate this. One is an academic problem, the other a practical problem. In Example 4.1, we relax assumptions A1 and A3 in Table 4.1. In Example 4.2, we relax all three assumptions.

4.1 Example: When the observation noise contains normal distribution

Consider a two-objective optimization problem $\min_{\theta\in\Theta} J(\theta)$, where $J(\theta)=[J_1(\theta), J_2(\theta)]^\tau$, $J_1(\theta)$ and $J_2(\theta)$ are the true performance of the design θ, and τ denotes transposition. There are 1000 designs in Θ. For each design θ, $J_1(\theta)$ and $J_2(\theta)$ are uniformly generated values from [0,1] and are fixed in the following experiments. The true performances are shown in Fig. 4.6.

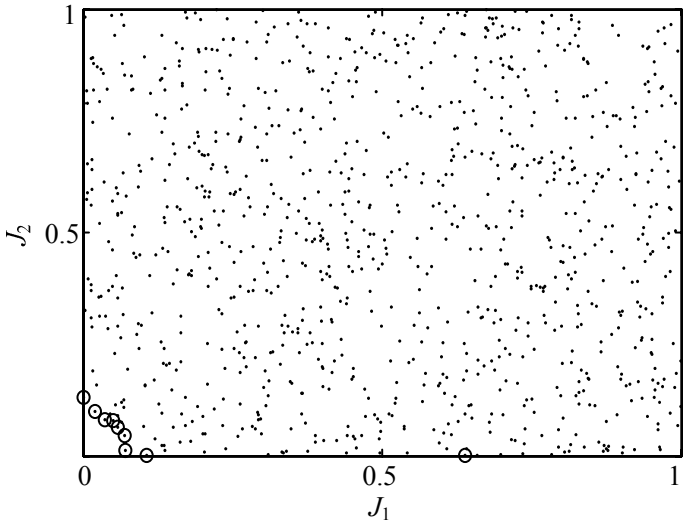


Fig. 4.6. True performances J_1 and J_2 in Example 4.1

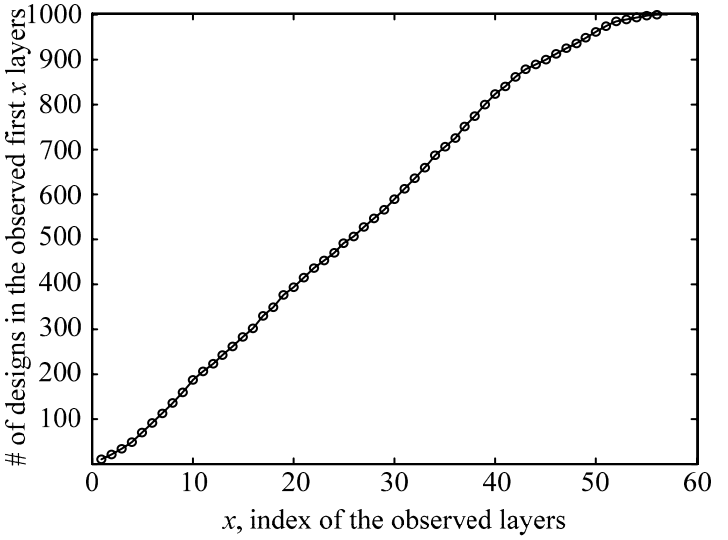


Fig. 4.7. The observed VOPC of Example 4.1

There are 9 designs in the Pareto frontier, which are marked by circles. The observation noise of each design is independent and has normal distribution $N(0, 0.252)$. We are interested to find at least k, $1 \le k \le 9$, designs in the true Pareto frontier with high probability, $\alpha \ge 0.95$. The question is how many observed layers we should select. In the following, two methods are compared.

First, we use the regressed values in Table 4.1 to answer this question. We simulate each design only once and estimate the VOPC type of this problem, which is neutral (as shown in Fig. 4.7). We specify the noise level as 0.5. Then, from Table 4.1, we find the values of coefficients as

$$Z_1=0.2176,\ Z_2=0.9430,\ Z_3=0.9208,\ Z_4=1.0479.$$

Since there are only 1000 designs and 579 observed layers in Example 4.1, we need to adjust the values of g and k. We keep the ratios g/N_l, k/N, s/N_l as constant, where N_l is the total number of true layers. We use the total number of observed layers $\hat{N}_l$ as an estimate of N_l. Then, we have

$$g' = \lfloor (100/57)\times 1 \rfloor = 1,$$
$$k' = (10000/1000)\times k = 10k,\ 1 \le k \le 9,$$

where $\lfloor a \rfloor$ represents the smallest integer that is not smaller than a. Using Eq. (1), we get $s'(k', g')$ and $s = \lceil (57/100)\times s \rceil$, where $\lceil a \rceil$ represents the

largest integer that is not larger than a. The predicted values of s are collected in Table 4.2 (denoted as $\hat{s}$).

Second, we use the experiments to estimate how many observed layers should be selected. We use 1000 independent experiments to estimate the AP of each (s, k). In this way, for different k, we get estimates of how many observed layers are enough so that the corresponding AP is no less than 0.95. We regard these estimates as true values and also list them in Table 4.2 (denoted by s^*).

In Table 4.2, we can see that the predicted values $\hat{s}$ based on the regressed model are always an upper bound of the true values s^*. If we want to find at least one design in the true Pareto frontier, it is sufficient to focus on the observed first 5 layers. There are only 78 designs on the average in these layers (a saving from 1000 to 78). Also note that, if we want to contain most or all (nine) designs in the true Pareto frontier, we still need to explore many designs. However, this is due to the fact that the noises are large in our example and it is not compatible with goal softening to insist on k=9.

Table 4.2. Predicted and true values of s for Example 4.1

k	s^*	$\hat{s}$	$\left\lvert\bigcup_{i=1}^{\hat{s}} \mathcal{L}_i\right\rvert$
1	3	5	78
2	5	9	166
3	7	12	241
4	9	16	340
5	11	19	420
6	14	23	517
7	17	26	596
8	22	30	692
9	32	33	756

4.2 Example: The buffer allocation problem

We will consider a 10-node queuing network, as shown in Fig. 4.8. In fact, this example has already been introduced in Section III.3, but we considered only one objective function then. Now we are going to consider two objective functions (introduction follows) here. Let us briefly review the problem formulation. There are two classes of customers with different arrival distributions (exponential and uniform distributions). Both classes arrive at any of the 0–3 nodes and leave the network after finishing all three stages of services. The routing is class dependent and is deterministic.

The buffer size at each node is finite and is the parameter that we can design. We say that a buffer is full if there are as many customers as that buffer size, not including the customer being served. Nodes 8-9 have individual queues but share one server. This network can model a large number of real-world systems, such as manufacturing, communication, and traffic network. We consider the problem of allocating 22 buffer units among the 10 nodes. We use B_i to denote the buffer size at node i, $B_i \geq 0$. For symmetry reasons, we require

$$B_0=B_1=B_2=B_3,\ B_4=B_6,\ B_5=B_7,\ B_8,\ B_9>0. \tag{4.6}$$

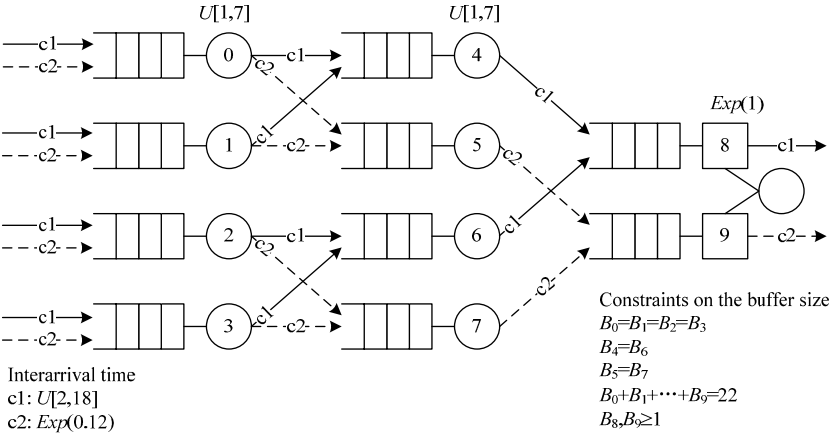


Fig. 4.8. The 10-node network with priority and shared server

We can get 1001 different configurations in all. There are two objective functions. One is the expected time to process the first 100 jobs from the same initial state (all buffers are empty). The other is the average utility of the buffers at all the nodes, i.e., $\sum_{i=0}^{9} q_i / B_i$, where q_i is the expected queue length at node i, $0 \leq i \leq 9$, where for $B_i = 0$, we define the utility of that buffer to be 1. We want to improve the throughput of the network and improve the efficiency of all the buffers. We formulate the problem as a two-objective minimization problem, where J_1 is the first objective function above and $J_2 = 1 - \sum_{i=0}^{9} q_i / B_i$.

For each design (a configuration of buffers) θ, we use 1000 independent experiments to estimate $J_1(\theta)$ and $J_2(\theta)$. The experimental results are shown in Fig. 4.9. We regard these values as true performances and define the configurations in the observed first two layers as good enough (9

designs in total), also marked by circles in Fig. 4.9. We want to find at least k, $1 \le k \le 9$, configurations in the true first two layers with high probability, $\alpha \ge 0.95$. The question is also how many observed layers should be selected.

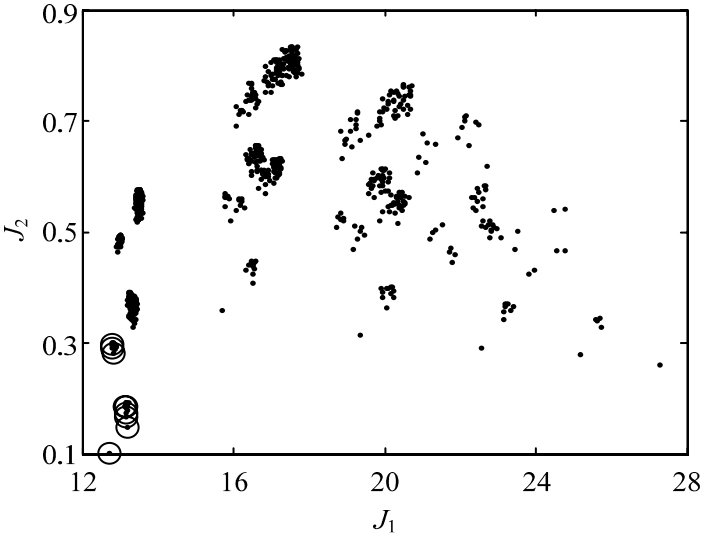


Fig. 4.9. The true performance of the configurations in Example 4.2

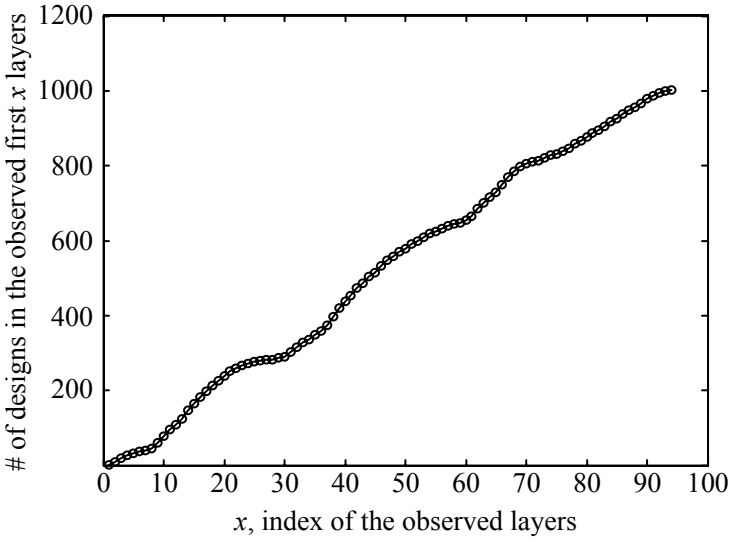


Fig. 4.10. The observed VOPC of Example 4.2

First, we simulate each configuration once (i.e., one replication only – a very crude estimate of the performance of the particular configuration). We show one instance in Fig. 4.10. There are 94 observed layers, which may be different in various experiments. The estimated VOPC type is neutral. By normalization, the standard deviation of the observation noise is 0.1017 for J_1 and 0.0271 for J_2, and we choose $0.5>2\times0.1017$ as the noise level. The corresponding coefficients in Table 4.1 are the same as those in Example 4.1. We adjust the values of g and k accordingly, i.e.,

$$g' = \lfloor 100/94\times 2 \rfloor = 2,\ k' = (10000/1001)\times k \approx 10k,\ 1 \le k \le 9.$$

Substituting these values into Eq. (4.4), we can get $s'=Z(k',\ g')$ and $s = \lceil 94/100\times s' \rceil$. We show the predicted number of observed layers to select in Table 4.3.

Second, we use 1000 independent experiments to estimate the AP of each $(s,\ k)$. For each k, when AP is no smaller than 0.95, we denote the value of s as s^* in Table 4.3.

Table 4.3. Predicted and true values of s for Example 4.2

k	s^*	$\hat{s}$	$\left\lvert \bigcup_{i=1}^{\hat{s}} \mathcal{L}_i \right\rvert$
1	1	5	32
2	2	8	44
3	2	11	95
4	3	14	147
5	4	17	197
6	5	20	240
7	6	23	268
8	8	26	279
9	9	29	286

If we want to contain at least 3 designs in the true first two layers, according to Table 4.3, we need to explore only 95 designs on average. This saves much of our search efforts. We can see that the predicted values of s are always no less than the estimated values. The predicted values of $\hat{s}$ seem conservative. The reason is that the normalized noise level is 0.2034 in Example 4.2, which means that some good enough designs almost always dominate some other designs in observation. However, the smallest noise level in Table 4.1 is 0.5, which is more than twice as large as the normalized. In turn, this leads to a conservative estimate of s.

Example 4.2 violates all the three assumptions: There are only 1001 designs (configurations) in total; the observation noise is not i.i.d., and

does not contain uniform distribution. However, numerical results show that the regression function (0.4) and the regressed coefficients in Table 4.1 still give good guidance on how many observed layers should be selected. When there are more objective functions, we can design similar experiments to obtain the regressed values of the coefficients in Eq. (4.4). In practical applications, we can use more problem information to obtain better prediction on the number of observed layers to select (i.e., a tighter upper bound of the true values). This will be shown in Chapter VIII Section 3.

Exercise 4.6: In Example 4.2, we introduced constraints in Eq. (4.6) for symmetry reason. What if we relax these constraints? Can we find designs with better performances in both objective functions? Please try to apply VOO to solve Example 4.2 without the constraints in Eq. (4.6).

Exercise 4.7: Please think, before reading Section VIII.3, about effective ways to obtain less conservative estimate of the number of observed layers to select, when we have the observed performance of the randomly sampled N designs.

Box 4.1. The application procedure of VOO

Step 1:	Uniformly and randomly sample N designs from Θ.
Step 2:	Use a crude and computationally fast model to estimate the m performance criteria of these N designs.
Step 3:	Estimate the VOPC class and noise level. The user specifies the size of good enough set, g, and the required alignment level, k.
Step 4:	Use Table 4.1 (if m=2, or similar tables generated beforehand for general m) to calculate s=Z(g, k/VOPC class, noise level).
Step 5:	Select the observed first s layers as the selected set.
Step 6:	The theory of VOO ensures that S contains at least k truly good enough designs with probability no less than 0.95.