Chapter III Comparison of Selection Rules

In Chapter II, we learned that the selected set, *S*, plays an important role in the theory of OO. We also introduced two natural ways to determine the set *S*, namely, the Blind Pick (BP) and the Horse Race (HR) method. Of course there are also other ways possible. We have already mentioned in Section II.2 the example of how we take inspiration from the tennis tournament, where the pair wise elimination rule is used to determine the 16 surviving players (i.e., the set *S*) for the quarter final of the U.S. Open. Other rules for picking *S* come to our mind inspired by sports tournaments, such as the round-robin comparison imitating baseball (Jia et al. 2006a), which will be defined formally in Section 1. It is equally intuitive that different selection rules will lead to different Alignment Probability, Prob $[|G\cap S|\geq k]$, all other parameters remaining the same. In other words, to achieve the same high alignment probability, different selection rules may require us to select different number of designs for *S*. After we apply OO to find out the selected set *S*, which contains some good enough designs with high probability, we still need to use the detailed model to accurately evaluate the performance of the designs in *S* and finally choose the best one. It is clear that the larger the selected set *S* is, the more simulation is needed in the post selection comparison. So, it is of practical importance to find out which selection rule requires the smallest selection set in a given optimization problem. This selection rule will save the computational effort of the detail simulation the most efficiently. The purpose of this chapter is to make a careful study of the efficiency of different selection rules in terms of computing burden and efficacy in achieving high values of AP. For readers not interested in details and only concerned with the practical use of OO, s/he can go directly to the general recommendation at the end of this chapter which concludes that the HR is a generally good selection rule in the absence of detailed analysis and the BP rule can of course always serve as a lower bound in performance.

Since the size of the selected set is of the most practical importance, we use the size to evaluate and compare the performance of the selection rules. We will consider selection rule A better than B if selection rule A requires a smaller selected set *S* than B to achieve the same AP, all other parameters remaining the same. A natural question is: Is there a unique selection rule that is better than any others? If the answer is yes, then we only need to use this universal best selection rule. This is of course convenient for practical application of OO. Unfortunately, we have not found such a selection rule yet. Instead, each selection rule may have special advantages in dealing with certain problems. For example, as an extreme case, Blind Pick needs no problem information, nor any performance evaluation (not even a rough estimate based on the crude model). However, compared with Horse Race, BP usually requires a larger selected set to ensure the same high AP. So, a reasonable way to evaluate the performance of different selection rules is to classify different scenarios first, such as the problem types, noise levels, size of good enough set, required alignment level, and the computing budget we can use during the selection. Then we will be able to figure out what is the best selection rule in each scenario.

We have tried the above ways to compare different selection rules. However, after we obtain all the comparison results, i.e., which selection rule is the best in each scenario, there arises another question, that is, how can we present these results in an easy form? Obviously, the first idea might be using a lookup table. In other words, we list the scenarios and the corresponding selection rules that have the best performance within that scenario, and do this listing for all the scenarios. But there are too many scenarios. To get a rough estimate of how long this table will be, let us consider the following numerical example. We know there are 5 types of OPCs, and 3 noise levels (small, middle, and large). Suppose we consider 3 computing budgets (small, middle, and large), 20 different sizes of the good enough set (top-1%, 2%,…,20% designs), and 10 different alignment levels ($k = 1,2,\ldots 10$). Then the total number of scenario would be: $5\times3\times3\times 20\times10 = 9000$. Suppose we have a table with 9000 lines, and the problem that we want to apply OO to solve fits in only one line, it would be inconvenient to find the line that is useful. We have to find better ways to present the comparison results.

We are fortunate to find this easy way to present the comparison results. The idea is to use a function with a few parameters to approximate the size of the selected set. Then giving the scenario, to which the practical problem belongs, and a selection rule, we use the function to approximate the size of the selected subset thus required. We do this for each selection rule. By comparing these approximated selection sizes, we then identify a predicted best selection rule in that scenario. As long as that prediction is good, i.e., either it is one of the several truly best selection rules in that scenario, or it is close to the truly best selection rule and the difference between the selected sizes is not large, then this predicted best selection rule is a truly good enough selection rule to be easily used in practice. We found such functions for many selection rules, including the ones that have already been frequently used in OO and some new ones. And we use two examples to show how this helps to find a good enough selection rule, and thus further save the computing budget.

The selection rules differ from each other considering how the computing budgets are allocated among the designs, how the designs are evaluated, and how the designs are finally selected. We should realize that there are a huge number of different selection rules. Most of these selection rules do not have a name yet, but they are different from the well-known rules such as BP and HR in the above sense. To get a rough idea, suppose there are all together $N = 1000$ designs, and we can evaluate these designs by at most $T = 30000$ simulations in total before selecting the subset. This equals to allocate these 30000 simulations to the 1000 designs. The number of all the allocation is

$$\begin{bmatrix} T+N-1 \\ N-1 \end{bmatrix} = \begin{bmatrix} 30999 \\ 999 \end{bmatrix} \gg 10^{1000},$$

which is a huge number. This example shows us two points. First, it is impractical to list and compare all the selection rules. Second, the selection rules we know is only a small portion of all the selection rule. So it is important to find some general properties, say selection rules with these properties are better (or have a large chance to be better) than others. These properties will help us to search for better selection rules in the future. We have found some these properties through theoretical analysis (Jia et al. 2006a; Jia 2006) and experimental study. One of them has justified that HR is in general a good selection rule.

The rest of this chapter is organized as follows. In Section 1, we classify the selection rules (especially the selection rules that are frequently used in OO literature) into different classes. Under mild assumptions, we establish a basic property of "good" selection rules. In Section 2, we use the selection size, s, to quantify the efficiency of the selection rule, and introduce a regression function to approximate this size for each commonly used selection rule. Through numerical experiments, we find that the comparison of the efficiency based on the regression function is almost exactly the same as the comparison result based on extensive simulation experiments. This justifies that the regression function is a reasonable tool to compare the efficiency of the selection rules. In order to show the readers how they can use the regression function to find a good selection rule and further reduce the selection size in practical problems, in Section 3, we consider two examples: a real function example and a queuing network example. In Section 4, we discuss three properties of good selection rules, which will help us to find better selection rules in the future. We summarize the application procedure in Section 5 and give some simple, quick and dirty rules to choose a good selection rule without detailed calculation and analysis.

1 Classification of selection rules

Let us now have a review of how a selection rule works. First, there is a crude model of the designs, which is computationally fast. In the extreme case of Blind Pick, no crude model is used. From an engineering viewpoint, this can also be regarded as using a crude model with infinite observation noise. So any observed order among the designs happens with equal probability. Second, based on the rough performance estimate thus obtained, a selection rule compares and orders the designs, and finally selects some designs as the selected set. Because the accuracy of the crude model used in the first step affects the size of the selected set dramatically, in the comparison of selection rules in this chapter, we assume the selection rules use the same crude model. Except in BP, a crude model with infinite observation noise is used. Note that different selection rules may use the rough performance estimate obtained through the same crude model in different ways. For example, a selection rule may compare designs in pairs, like in the U.S. tennis open, and regards all the designs in the quarter final as observed good enough designs. These 16 designs may be different from the observed top-16 designs, if selected by the Horse Race, because the observed second design may compete with the observed best design in an early round and fail, and thus cannot move into the quarter final. Also note that different designs may receive a different number of observations during the selection procedure. For example, in a season in National Basketball Association (NBA), each team plays the same number of games. If we regard the outcome of each game as an observation on the performances of the teams, each team receives the same number of observations. However, in the U.S. tennis open, if a player fails in an early round, he/she does not have the chance to move into the next round, which means finally the player may receive a different number of observations. Obviously, in simulation-based optimization, each observation of a design takes some part of the computing budget. That means different selection rules may assign the computing budget to the designs in different ways.

In summary, each selection rule in OO must answer two questions:

1. How to select S? – by ordering all N designs via either (a) or (b) below and select the top-s.
   (a) Using the estimated cardinal value no matter how crude it is.

(b) By the number of "wins" accumulated from the comparison of the estimated cardinal values no matter how crude they are. Two examples of the comparisons are:
   (i) Comparison done pair-wisely
   (ii) Comparison done globally.

2. How much computing budget is assigned to a design? – by either (a) or (b).
   (a) Predetermined and allocated once.
   (b) Successive iteration after initial assignment
      (i) With elimination (i.e., some design will receive no more computing budget after certain iteration)
      (ii) Without elimination.

Using answers to the above two questions, we consider and classify the following selection rules that are frequently used in ordinal optimization.

- **Blind pick (BP):** Assumes no knowledge of the problem and uniformly pick up *s* designs to make up *S*, i.e., (Question 1) random pick and (Question 2) no budget assigned (predetermined).
- **Horse race (HR):** The numbers of independent observations allocated to all designs are equal. By comparing the sample average, the observed top-*s* designs are selected. (Question 1) (a) and (Question 2) (a).
- **Round robin (RR):** Every design compares with every other design pair-wisely. In each comparison, we use only one observation (or equivalently "replication" in simulation language) per design to estimate the performance. Upon we completing the comparisons, every design wins some number of comparisons, including zero. We sort the designs by the number of wins in decreasing order[1]. The first-*s* designs are selected. For example, if there are four designs: θ_1, θ_2, θ_3, and θ_4. We need 3 rounds of comparisons:

Round 1: θ_1 vs. θ_2, θ_3 vs. θ_4.
Round 2: θ_1 vs. θ_3, θ_2 vs. θ_4.
Round 3: θ_1 vs. θ_4, θ_2 vs. θ_3.

Assume θ_1 and θ_3 win in round 1; θ_1 and θ_2 win in round 2; θ_1 and θ_2 win in round 3. Then the four designs win 3,2,1, and 0 comparisons,

[1] When a tie appears, the orders of designs within a tie are randomly decided. This assumption is also held for all the other selection rules mentioned in this book.

respectively. The observed best design is θ_1 and the worst is θ_4. (Question 1) (b)-(i) and (Question 2) (b)-(ii).

- **Sequential pair-wise elimination (SPE):** This is the rule used in tennis[2]. Designs are initially grouped into many pairs. The winners of these pairs are grouped into pairs again. This continues until a final winner appears. We show one example in Fig. 3.1. Assume θ_1 and θ_4 win in round 1; θ_1 wins in round 2. Then θ_1 is the observed best design. (Question 1) (b)-(i) and (Question 2) (b)-(i).

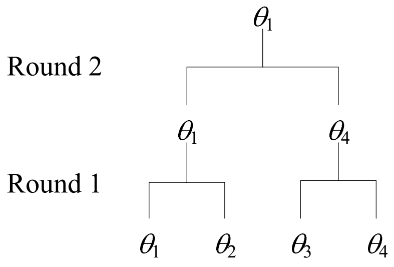


Fig. 3.1. One example of sequential pair-wise elimination (Jia et al. 2006a) © 2006 Elsevier

- **Optimal Computing Budget Allocation (OCBA)** (Chen et al. 2000)**:** The idea is that we want to lavish larger computing budget on designs that more likely turn out to be good enough and not to waste efforts on designs that have a high likelihood of being bad. First, we randomly sample m_0 designs from Θ, and take n_0 observations of each design. Then we use a formula (Chen et al. 2000) to allocate additional Δ computing budget units to these m_0 designs to perform more replications. This procedure continues until all the computing budget is consumed. In OCBA we fix the "breadth"[3]. Section VII.4 has more details on this method. (Question 1) (a) and (Question 2) (b)-(ii).
- **Breadth vs. depth (B vs. D)** (Lin 2000b): The idea is to always allocate the next Δ computing budget units in the way that leads to a greater marginal benefit. There are two ways to get a marginal benefit: The breadth process, which means to sample new designs, and the depth

[2] In tennis tournament, the initial pairing is actually not totally random. We shall using total random pairing here since we assume no prior information on the designs.

[3] The "breadth" represents the number of designs explored in the optimization, and the "depth" represents the number of computing budget units allocated to each designs.

process[4], which means to do more observations of the designs that have already been sampled in order to get a better estimate of the design performance. The "benefit" of considering both breadth and depth is to increase the possibility of including truly good enough designs in the selected set. In B vs. D we can change the "breadth." (Question 1)(a) and (Question 2)(b)-(ii).

- **HR with global comparison (HR_gc):** In every round, the winners from the last round each receive one computing budget unit, and are compared with each other based on the new observations only. The observed best half designs win and the losers are eliminated in successive rounds. Finally we sort the designs by the number of rounds each design enters, from the largest to the smallest. (Question 1)(b)-(ii) and (Question 2)(b)-(i).
- **HR with no elimination (HR_ne):** In round i we compare the mean values of the observations so far, and allocate the additional Δ_i computing budget units to the observed best m_i designs. The value of Δ_i and m_i reduce by half each time. (Question 1)(a) and (Question 2)(b)-(ii).
- **HR as a counterpart of round robin (HR_CRR):** We allocate the computing budget as in RR. Finally we sort the designs by the average value of the observed data, not the number of wins. (Question 1)(a) and (Question 2)(b)-(ii).

We summarize the different rules in Table 3.1 below.

Table 3.1. Classification of selection rules

Selection Rule	How to select S?	Computing budget
BP	Random picking	No computing budget needed
HR	A	a
RR	b-i	b-ii
SPE	b-i	b-i
OCBA	A	b-ii
B vs. D	A	b-ii
HR_gc	b-ii	b-i
HR_ne	A	b-ii
HR_CRR	A	b-ii

Comparison and analysis in Section 2 and 3 will demonstrate that HR is in general a good selection rule, which is also a reason why we consider so

[4] For this rule we use BP in the breadth process and OCBA in the depth process. Other choices are possible and will yield different results (Lin 2000b).

many variants of the HR selection rules above (such as HR_ne, HR_gc, and HR_CRR).

Before going into details, we show here a basic property common to all comparisons of selection rules. Colloquially, we state it as

> ***Seeing is believing*** – everything being equal (i.e., the same computing budget allocated to each design), we should always choose the observed top-s designs as the selected set.

First we introduce some notations. We use $\hat{J}(\theta_i)$ to denote one observation of design θ_i, i.e.,

$$\hat{J}(\theta_i) = J(\theta_i) + w_i = L(x(t;\theta_i,\xi)), \tag{3.1}$$

where w_i denotes the observation noise and $J(\theta_i)$ is the true performance value of design θ_i. We use $\bar{\hat{J}}(\theta_i)$ to denote the average observed value of the performance of design θ_i based on N_i independent observations (replications), i.e.,

$$\bar{\hat{J}}(\theta_i) = \frac{1}{N_i}\sum_{j=1}^{N_i} L\left(x\left(t;\theta_i,\xi_j\right)\right) \tag{3.2}$$

where ξ_j is the randomness in the j-th sample replication. To simplify the discussion, we introduce the following assumptions:

Assumption 1: Observation noises of different designs are independently and identically distributed, i.e., w_i is i.i.d. for i=1,2,…N.
Assumption 2: We have no prior knowledge of the designs before conducting the simulations.

Now we state formally a basic property of the optimal selection rules.

Basic Property: Under Assumptions 1 and 2 and the equal allocation of computing budget to all designs, the selection of the observed top-s designs (by whatever rule) leads to an alignment probability no less than other selections on the average.

This property is intuitively reasonable and will be proved below. We give a short illustration in a specific case first to lend insight. Assume that there are only two designs θ_1 and θ_2 with true performances $J(\theta_1) < J(\theta_2)$, each design is observed only once, and the observation noises w_1 and w_2 contains i.i.d distribution. We define the truly better design θ_1 as the good enough design. And we can select only one design based on the observed performances. The alignment probability is simplified to the probability that we correctly select the better design θ_1. The question is: which design shall we select? In this case, the Basic Property says we should select the observed better one. To see why this is correct, we compare the probability that θ_1 is observed better $\text{Prob}\left[\bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_2)\right]$ and the probability that θ_2 is observed better $\text{Prob}\left[\bar{\hat{J}}(\theta_1) > \bar{\hat{J}}(\theta_2)\right]$. For the first probability, we have

$$\text{Prob}\left[\bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_2)\right] = \text{Prob}\left[J(\theta_1) + w_1 < J(\theta_2) + w_2\right]$$
$$= \text{Prob}\left[w_1 - w_2 < J(\theta_2) - J(\theta_1)\right].$$

Similarly, we can rewrite the second probability as

$$\text{Prob}\left[\bar{\hat{J}}(\theta_1) > \bar{\hat{J}}(\theta_2)\right] = \text{Prob}\left[w_1 - w_2 > J(\theta_2) - J(\theta_1)\right].$$

Since w_1 and w_2 are i.i.d., w_1-w_2 is with zero mean and has symmetric probability density function. Then due to the fact that $J(\theta_2)$-$J(\theta_1)$>0, it is obvious that

$$\text{Prob}\left[w_1 - w_2 < J(\theta_2) - J(\theta_1)\right] \geq \text{Prob}\left[w_1 - w_2 > J(\theta_2) - J(\theta_1)\right],$$

which means

$$\text{Prob}\left[\bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_2)\right] \geq \text{Prob}\left[\bar{\hat{J}}(\theta_1) > \bar{\hat{J}}(\theta_2)\right], \tag{3.3}$$

i.e., the truly better design has a larger chance to be better observed. If we follow the Basic Property, the alignment probability would be $\text{AP}_1\text{=Prob}\left[\bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_2)\right]$. If we do not follow the Basic Property, we must select the observe worse design with positive probability $q > 0$, then the alignment probability is

$$\begin{aligned}
&\mathrm{AP}_2\\
&=\mathrm{Prob}\left[\text{select } \theta_1, \bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]+\mathrm{Prob}\left[\text{select } \theta_1, \bar{\hat{J}}(\theta_1)>\bar{\hat{J}}(\theta_2)\right]\\
&=\mathrm{Prob}\left[\text{select } \theta_1 \middle| \bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]\\
&\quad+\mathrm{Prob}\left[\text{select } \theta_1 \middle| \bar{\hat{J}}(\theta_1)>\bar{\hat{J}}(\theta_2)\right]\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)>\bar{\hat{J}}(\theta_2)\right]\\
&=(1-q)\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]+q\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)>\bar{\hat{J}}(\theta_2)\right]\\
&\le(1-q)\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]+q\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]\\
&=\mathrm{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2)\right]\\
&=\mathrm{AP}_1.
\end{aligned}$$

This means the Basic Property leads to an alignment probability no less than other selections. This also means "to see is to believe". When both designs are observed $N_1=N_2$ times, since the noise of a single observation is i.i.d. for both designs, the difference between the average observed performance $\bar{\hat{J}}(\theta_i)$ and the true performance $J(\theta_i)$ (i.e., $\bar{\hat{J}}(\theta_i)-J(\theta_i)$) is also i.i.d. for both designs. Then we can extend the above proof to this case straightforwardly.

Exercise 3.1: Please extend the above proof to the case when both designs are observed more than once but equal times, i.e., $N_1 = N_2>1$.

We can use proof by contradiction to show the Basic Property in more general cases. First, note that when there are more than two designs, Eq. (3.3) still holds for any two designs (then the θ_1 in Eq. (3.3) represents the truly better one of the two designs). Suppose the selected set *S,* on the contrary, is not the observed top-*s* designs, then there must be some design θ that is within the observed top-*s*, but not selected; and there must be some other design θ' that is not within observed top-*s*, but is selected. Now we construct another selected set *S*'. The only difference between *S* and *S*' is: θ is in *S*' but θ' is not. Following Eq. (3.3), we know a truly better design θ has a bigger chance to be observed good enough. So we have

$$\mathrm{Prob}\left[|S\cap G|\ge k\right]\le\mathrm{Prob}\left[|S'\cap G|\ge k\right]. \tag{3.4}$$

Following Eq. (3.4), each time we add to the selected set a design that is within the observed top-s and remove a design that is not, the AP does not decrease but only increase. If we keep on doing this, eventually we can obtain the selected set containing exact observed top-s designs whose alignment probability is no less than $\text{Prob}\left[|S\cap G|\geq k\right]$. This proves the Basic Property.

Exercise 3.2: when will the inequality in Eq. (3.4) be strict, and when not?

Astute reader might wonder: Is Assumption 1 critical to the Basic Property? The answer is yes. We show in the following an example, in which the observation noise is not i.i.d., and the Basic Property does not hold any more. Suppose there are three designs, θ_1, θ_2, and θ_3, with true performances $J(\theta_1)=0$, $J(\theta_2)=1$, and $J(\theta_3)=1.0001$. The observation noise is independent but contains non-identical normal distribution such that

$$\bar{\hat{J}}(\theta_1)\sim N\left(J(\theta_1),\sigma_1^2\right),$$
$$\bar{\hat{J}}(\theta_2)\sim N\left(J(\theta_2),\sigma_2^2\right), \text{ and}$$
$$\bar{\hat{J}}(\theta_3)\sim N\left(J(\theta_3),\sigma_3^2\right).$$

Let $\sigma_1 = 0.0001$, $\sigma_2 = \sigma_3 = 10$. Then we have

$$\begin{aligned}&\text{Prob}\left[\theta_1 \text{ is observed as the best}\right]\\&=\text{Prob}\left[\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_2),\bar{\hat{J}}(\theta_1)<\bar{\hat{J}}(\theta_3)\right]\\&=\int_{-\infty}^{+\infty}\text{Prob}\left[\bar{\hat{J}}(\theta_2)\geq x\right]\text{Prob}\left[\bar{\hat{J}}(\theta_3)\geq x\right]p\left(\bar{\hat{J}}(\theta_1)=x\right)dx.\end{aligned}$$

The last integration can be calculated using numerical integration. Similarly, we can calculate the probability that θ_2 or θ_3 is observed as the best. If we calculate the probability that each design is observed as the best, then we have

$$\begin{aligned}&\text{Prob}[\theta_1 \text{ is observed as the best}]\approx 0.2914,\\&\text{Prob}[\theta_2 \text{ is observed as the best}]\approx 0.3543,\\&\text{Prob}[\theta_3 \text{ is observed as the best}]\approx 0.3543.\end{aligned}$$

We can similarly calculate

$$
\begin{aligned}
&\text{Prob}\left[\theta_1 \text{ is observed as the middle}\right] \\
&=\text{Prob}\left[\bar{\hat{J}}(\theta_2) < \bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_3)\right] + \text{Prob}\left[\bar{\hat{J}}(\theta_3) < \bar{\hat{J}}(\theta_1) < \bar{\hat{J}}(\theta_2)\right] \\
&= \int_{-\infty}^{+\infty} \text{Prob}\left[\bar{\hat{J}}(\theta_2) < x\right] \text{Prob}\left[\bar{\hat{J}}(\theta_3) > x\right] p\left(\bar{\hat{J}}(\theta_1) = x\right) dx \\
&\quad + \int_{-\infty}^{+\infty} \text{Prob}\left[\bar{\hat{J}}(\theta\ ) > x\right] \text{Prob}\left[\bar{\hat{J}}(\theta_3) < x\right] p\left(\bar{\hat{J}}(\theta_1) = x\right) dx.
\end{aligned}
$$

The numerical results are

Prob[θ_1 is observed as the middle]≈0.4968,
Prob[θ_2 is observed as the middle]≈0.2516,
Prob[θ_3 is observed as the middle]≈0.2516.

In this case, if we follow the Basic Property and select the observed best design each time, then the alignment probability is 0.2914. However, if we select the observed middle design each time, then the alignment Probability is 0.4968. This means the Basic Property leads to a smaller alignment probability. This constitutes a counterexample if Assumption 1 is relaxed.[5]

Based on the Basic Property, in the following, our discussion will be based on Assumption 3 as we proceed below, if not mentioned explicitly:

Assumption 3. No matter what measurement and comparison method used, in every selection rule, the observed top-|*S*| designs (under that specific measurement and comparison method) will be finally selected to compose set *S*.

We shall now compare selection rules in the following way:

[5]However, we should not take this counter example too seriously since the parameters used are most extreme. Almost all the analysis associated with OO in this book are very conservative in nature. Empirically in numerous applications, we and others have found that OO tools work well despite theoretical conditions being violated or impossible to verify. As we shall see later, in selection rules such as OCBA or B vs. D, assumption 1 is not satisfied. Noises are independent but not identical. Yet, in no cases, the analysis and prediction of this chapter did not work empirically.

Everything being equal (i.e., the same optimization problem, observation noise, computing budget, G, k, and α), which selection rule γ can use the smallest S to make $\text{Prob}\left[|G \cap S| \geq k/\gamma\right] \geq \alpha$ ***?***

2 Quantify the efficiency of selection rules

From the viewpoint of application, we may see the contribution of OO in this way: After screening (with or without iteration) using a crude model, a user is most interested in which of the subset of designs thus explored s/he should lavish her/his attention on in order to obtain a truly good enough design with sufficiently high probability. This selected subset denoted as *S* ideally should be as small as possible to minimize search and computing efforts. As we mentioned at the beginning of this chapter, we need to figure out what factors affect the size of the selected set, and by classifying all the scenarios, we compare the selection rules in each scenario and list the best for application. For a given high probability, α, the required minimum size for *S* is a function of the following parameters:

- $|G|=g$ – the size of the good enough set, e.g., the top-*n*%
- *k* – the number of members of *G* guaranteed to be contained in *S*
- Noise/Error level, σ – in the estimation of the system performance using the crude model. We assume such noise is i.i.d. and can be roughly characterized as *small*, *medium*, and *large* with respect to the performance values being estimated. The case of correlated noise/error can also be handled (Deng et al. 1992), which will be discussed in Section VII.3.
- Problem Class, *C* – clearly if we are dealing with a problem contains many good enough solutions vs. a needle in a haystack type of problem, the required size of *S* can be very different. We express this notion in the form of an Ordered Performance Curve (OPC) which is by definition a monotonically nondecreasing curve that plots true performance value against all the designs ordered from the best to the worst (as defined in Chapter II and (Lau and Ho 1997; Ho et al. 1992; Ho 1999)). Five types of OPC are possible as illustrated in Fig. 3.2.
- Computing budget, *T* – the total number of simulations that can be used during the selection.

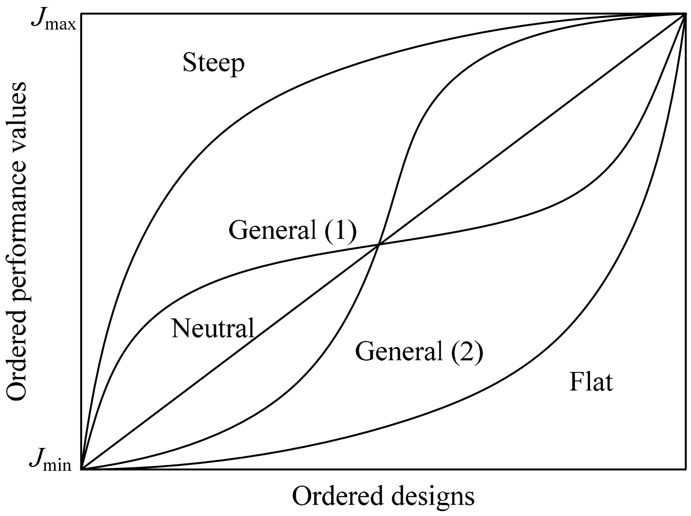


Fig. 3.2. Examples of ordered performance curves: flat, neutral, steep, and general

These factors can help distinguish a lot of different scenarios. As aforementioned, it is inconvenient for practitioners to use a very long lookup table to determine which selection rule is the best for a specific problem. So we need a function to approximate the size of the selected subset for each selection rule and in each scenario. In Section II.5, we introduced a regression function as follows:

$$Z(k,g) = e^{Z_1} k^{Z_2} g^{Z_3} + Z_4, \tag{3.5}$$

where Z_1, Z_2, Z_3, Z_4 are constants depending on the OPC class, the noise level, g, and k values. When using the Horse Race selection rule and each design takes only one observation, Eq. (3.5) approximates the size of the selected subset pretty well. To test whether Eq. (3.5) can also be used to approximate the size of the selected subset for other selection rules and in other scenarios, we compare the true and the approximated values in each of the 9000 scenarios, for each selection rule. The detailed parameter settings of the experiments will be introduced in section 2.1. To get the true value of the size of the selected set, the idea is similar to that in Section II.5: we fix the true performance of each design, which then represents one of the five OPCs. Then we use a selection rule γ to determine which designs to select. Each time when the selection rule requires observing the performance of a design θ, we randomly generate the observation noise w and use $J(\theta)+w$ as an observed performance. If the selection rule assigns different numbers of replications to different designs, then the noises contained in observed performances will be different. All the available computing budgets are utilized in the way that is specified by the selection rule, and finally the selection rule obtains an order of all the designs according to the observed performances. Then we check whether there are at least k truly top-g designs contained in the observed top-s designs as selected by the selection

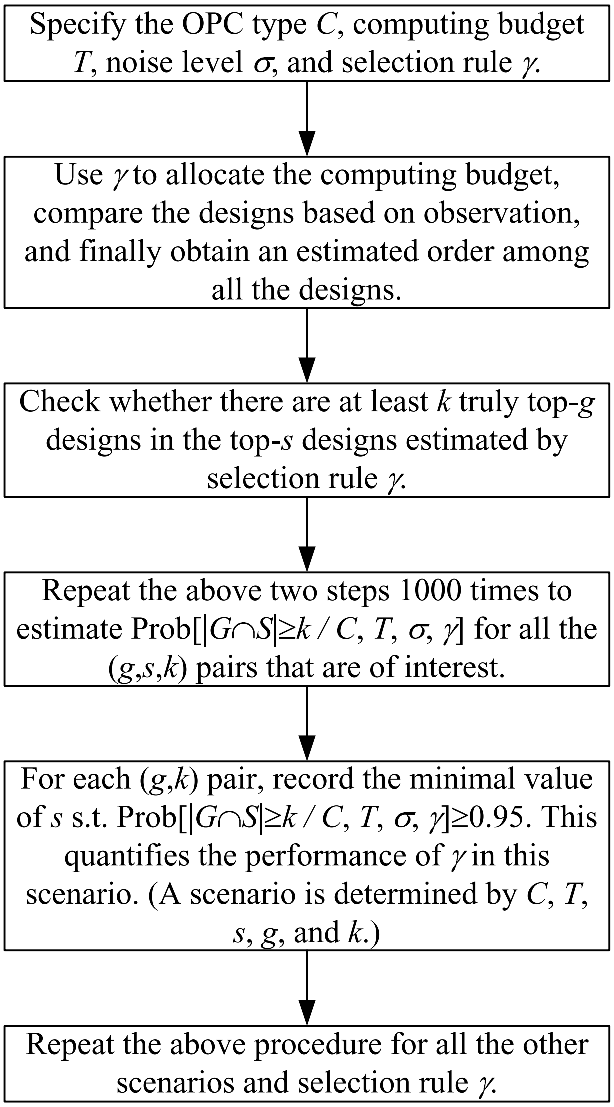

Specify the OPC type C, computing budget T, noise level σ, and selection rule γ.
Use γ to allocate the computing budget, compare the designs based on observation, and finally obtain an estimated order among all the designs.
Check whether there are at least k truly top-g designs in the top-s designs estimated by selection rule γ.
Repeat the above two steps 1000 times to estimate Prob[$|G \cap S| \geq k / C, T, \sigma, \gamma$] for all the ($g$,$s$,$k$) pairs that are of interest.
For each (g,k) pair, record the minimal value of s s.t. Prob[$|G \cap S| \geq k / C, T, \sigma, \gamma$]$\geq$0.95. This quantifies the performance of γ in this scenario. (A scenario is determined by C, T, s, g, and k.)
Repeat the above procedure for all the other scenarios and selection rule γ.

Fig. 3.3. The procedure to quantify the true performance of a selection rule

rule used. We check for each possible (g,s,k) values. Then we know whether a selection rule succeeds or fails in this experiment. We do 1000 experiments and then obtain the probability to succeed (as percent of the 1000 experiments) for each (g,s,k) values in that scenario. For each (g,k) pair, we record the minimal value of "s" s.t. the corresponding AP is no less than 0.95. This s is regarded as the true performance of selection rule γ in that scenario. The above procedure to obtain the true values of s in different scenarios is summarized in Fig. 3.3.

Giving these required values of s in different scenarios, we now want to find an easy way to represent these values instead of using a lookup table with 9000 rows. As aforementioned, we use Eq. (3.5) as the easy way to approximate these values of s in different scenarios. For all the (g,s,k) values with high enough AP, we regress the values of the coefficients in Eq. (3.5). Then for each (g,k) pair, the regressed value of s based on Eq. (3.5) is regarded as the approximate required value. We find this approximation is good in all the scenarios.

To get a rough idea, we show the examples of four selection rules (HR, OCBA, SPE, HR_ne) in the scenario of large computing budget (T=30000), neutral OPC, and middle noise level (σ= 0.5) in Fig. 3.4.

In Fig. 3.4, the dashed lines represent the regressed values based on Eq. (3.5). The solid lines represent the true values. The dashed lines approximate the solid lines well (in the trend and in the subset sizes of integer values of k). We list the maximal absolute difference between the solid and dashed lines in Table 3.2.

Table 3.2. The maximal absolute approximation error of required subset sizes (unit: number of designs) (Jia et al. 2006a) © 2006 Elsevier

g	HR	OCBA	SPE	HR_ne
10	3.8[6]	4.1	7.0	3.5
20	3.9	4.6	3.8	5.5
30	5.0	5.7	6.2	4.3
40	3.9	4.0	6.0	4.9

Since we see that the regression function in Eq. (3.5) is a good approximation to the true values of the selected subset size for each selection rule and in each scenario, we need to obtain the values of the coefficients in Eq. (3.5) for all the selection rules and for all the scenarios. Then giving a scenario, we can use Eq. (3.5) to predict which selection rule requires the

[6] The regressed subset size may not be integer, so the difference between the solid and dashed lines may not be integer. In application, we can just use the smallest integer that is no smaller than the regressed value as the subset size.

smallest subset to select. In Section 2.1, we introduce the parameter settings of all the scenarios in details. In Section 2.2, we discuss how we should compare selection rules using Eq. (3.5).

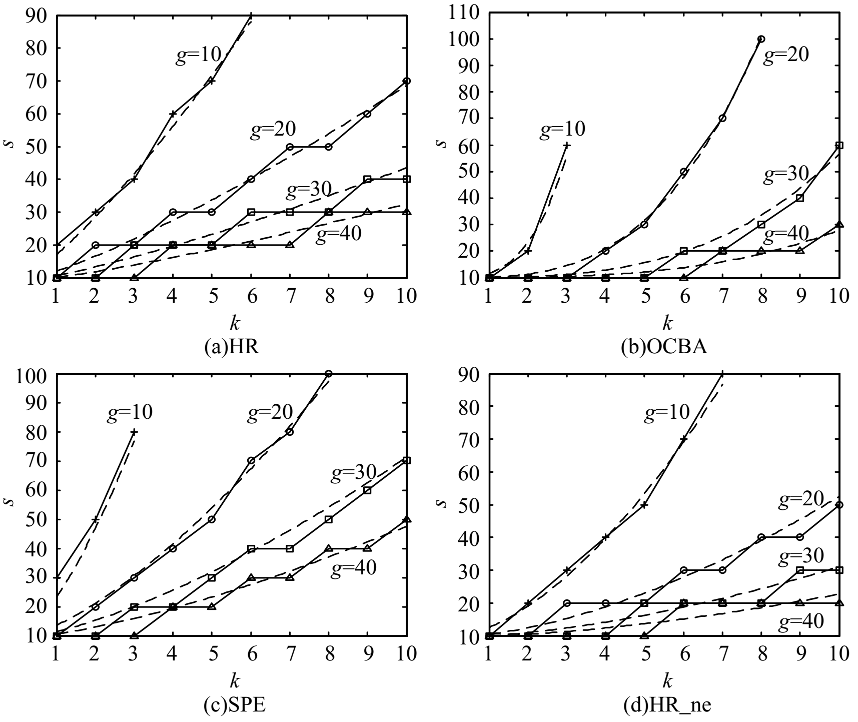


Fig. 3.4. The regressed values of Eq. (3.5) are good approximations of the true values of s. (Scenario: large computing budget, neutral OPC, and middle noise level) (subfigure (b) is from (Jia et al. 2006a) © 2006 Elsevier)

2.1 Parameter settings in experiments for regression functions

This subsection might be interesting only to readers who want to repeat our experiments. Other readers may want to skip this subsection and go directly to Section 2.2 and see how they can compare selection rules based on the regression functions.

In our experiments, we use the following parameter settings.

- $|\Theta|=N=1000$;
- $g\in\{10,20,\ldots200\}$;
- $s\in\{10,20,\ldots200\}$;
- $k\in\{1,2,\ldots10\}$;

- noise level: Assume the observation noise is independently and identically distributed (i.i.d.) and has normal distribution $N(0,\sigma^2)$, where $\sigma \in \{0.25, 0.5, 1.5\}$ for small, middle, and large noises respectively;
- Problem class: 5 OPC classes distinguished by different performance density functions.
  1. Flat: $J(\theta)=\theta^{10}$.
  2. U-Shaped: $J(\theta)=0.5sin(\pi(\theta\text{-}0.5))+0.5$.
  3. Neutral: $J(\theta)=\theta$.
  4. Bell: $J(\theta)=0.5\text{-}0.5(2\theta\text{-}1)^2$, $0\le\theta\le0.5$; $0.5+0.5(2\theta\text{-}1)^2$, $0.5\le\theta\le1$.
  5. Steep: $J(\theta)=1\text{-}(\theta\text{-}1)^{10}$, all (1)-(5) are defined on $\theta\in[0,1]$.

The OPCs and the corresponding performance densities are shown in Fig. 3.5.

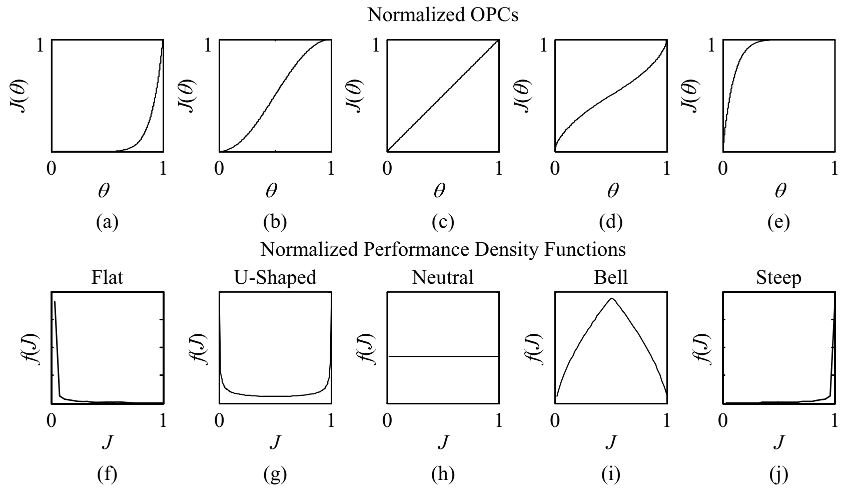


Fig. 3.5. Five types of OPCs and corresponding performance densities (Jia et al. 2006a) © 2006 Elsevier

- Computing budget: $T\in\{500,1000,30000\}$ for small, middle, and large computing budgets.
- "breadth" – the number of designs that can be explored by a selection rule. There is a close relationship between the "breadth" and the probability to find a good enough design. In one extreme case, a selection rule can maximize the "breadth" by observing each design only once. In this

way, the selection rule has a large chance to sample some truly good enough designs. However, since the observation noise is large, the selection rule may not be able to finally select these truly good enough designs. So a good selection rule should have a reasonable "breadth" and can adaptively change the "breadth" when the computing budget changes. The selection rules considered in this chapter require different computing budgets to explore the same number of designs (as shown in Table 3.3, where m_0 is the initial "breadth" and n_0 is the initial "depth" for each of these n_0 initial sampled designs. These formulas are obtained based on the definition of these selection rules in Section 1.). So when fixing the computing budget, the "breadth" of these selection rules are also different (as shown in Table 3.4).

- Other parameters: In OCBA and B vs. D, we try different values for n_0 and δ and find the following values works the best in the experiments, i.e., set $n_0 = 2$ and $\delta = 100$ when $T = 30000$; $n_0 = 1$ and $\delta = 10$ when $T = 1000$.
- For the combination of parameter settings above, we use 1000 independent experiments to estimate the alignment probability, which is expected to meet $\alpha \geq 0.95$.

Table 3.3. Computing budget units required by different selection rules to explore *N* designs (Jia et al. 2006a) © 2006 Elsevier

Selection rule	Computing budget units
BP	0
HR	N
OCBA	$\geq n_0 \times m_0$[7]
B vs. D	$\geq n_0 \times m_0$
SPE	$2N$-2
HR_gc	$2N$-2
HR_ne	$2N$-2
RR	N^2-N
HR_CRR	N^2-N

[7] m_0 and n_0 in OCBA and B vs. D are parameters controlled by the user. These two selection rules require $n_0 \times m_0$ computing budget units to roughly evaluate the m_0 randomly sampled designs at the beginning of the optimization, so the total required computing budget is at least $n_0 \times m_0$.

Table 3.4. The number of designs explored ("breadth") in different selection rules (Jia et al. 2006a) © 2006 Elsevier

Selection rule	Breadth		
	T=30000	T=1000	T=500
BP	1000	1000	500
HR	1000	1000	500
OCBA	1000[8]	500	400
B vs. D	≥10[9]	≥10	≥10
SPE	1000	501	251
HR_gc	1000	501	251
HR_ne	1000	501	251
RR	173	32	22
HR_CRR	173	32	22

Together with the blind pick selection rule, we tabulate all coefficients of the regression functions of different selection rules, as shown in Appendix C. It should be noted that since the subset sizes calculated by the coefficients above are used to approximate the experimental data, the values of g, s, and k should not exceed the above parameter settings of g, s, and k, which are used in the experiments, and the corresponding true AP Prob[$|G \cap S| \geq k$] should be no smaller than 0.95. To be specific, the approximation has a working range of $20 \leq g \leq 200$, $s=Z(\bullet/\bullet)<180$, and the values of k and g should let the fraction k/g be small[10,11]. On the occasions where the noise factor is characterized to be within these predetermined levels (i.e., σ takes other values than 0.25, 0.5, and 1.0), proper interpolation

[8] Since the "breadth" of OCBA is fixed and set by the user, we tried different values. The values shown here are the best ones found in the experiments.

[9] B vs. D automatically changes the "breadth" based on the observed performance. So we only show the lower bound here, which is m_0 the initial number of designs sampled from the design space. The value of m_0 is set by the user. So we tried different values of m_0, and showed the best ones here.

[10] For B vs. D we have more constraints on the working range. When T=30000, the working range should meet $1 \leq k \leq 5$ and $k/g \leq 1/15$. When T=1000, the working range should meet $1 \leq k \leq 3$ and $k/g \leq 1/25$. When T=500, the working range should meet $1 \leq k \leq 2$ and $k/g \leq 1/35$.

[11] For RR and HR_CRR, we only list the regression values in the case of large computing budget (T=30000). In this case, RR and HR_CRR can explore 173 designs, and still have many choices of selected subset sizes. But when T=1000 and 500, the two selection rules can explore only 32 and 22 designs accordingly. To cover 1 or 2 of top-100 designs with probability no smaller than 0.95, we usually need to select all the explored designs, the number of which are 32 and 22, respectively.

of the subset sizes will suffice. When we need to predict the size of the selected subset in a scenario for a selection rule, we first find the table of that scenario and for that selection rule in Appendix C, then obtain the coefficients for the variables in Eq. (3.5). Then we use Eq. (3.5) to predict the value of s. Based on this prediction, we can compare different selection rules and find the best one.

2.2 Comparison of selection rules

Using these regressed values in the last subsection, we can easily predict the most efficient selection rule (among all selection rules of concern) in each scenario. As we will show by numerical examples in this subsection, the predicted best selection rule is usually one of the top-n selection rules. In the following, we consider one scenario as an example: large computing budget, middle noise level, and neutral OPC class. First, we use experiments to compare HR, OCBA, and B vs. D, the three selection rules that are frequently used in OO literature. As mentioned at the beginning of this section just prior to Fig. 3.3, we regard the size of the selected subset obtained by the experiments as true values. So we will regard the comparison result based on the experimental results as the true result. This is shown in Fig. 3.6 and explained below.

Fig. 3.6 shows in each (g, k) values, which selection rule is the best. Let us take a closer look at this figure, and discuss whether these comparison results are reasonable. First, the squares appear where the good enough set and the required alignment level k are small. When the good enough set is defined as the best design, the alignment probability Prob[$|G \cap S| \geq k$] reduces to Prob[the observed best is the truly best], which is also known as the *probability of correct selection (PCS)* in OCBA. OCBA is developed to improve this probability. B vs. D is developed to improve Prob[the observed best design is truly good enough], which is close to PCS. Previous results in (Chen et al. 2000; Lin 2000b) also show that OCBA and B vs. D outperform HR in these cases.

Second, the black points appear where the good enough set is small, but k is large. To cover many good enough designs in the finally selected subset, we should make sure to explore a large number of good designs. Exploration, i.e., breadth, is thus more important than exploitation, i.e., depth. HR is developed to explore as many designs as possible, which is just the best choice in these cases. OCBA places emphasis on ensuring the observed best is the truly best, but does not pay attention to cover good enough designs. B vs. D takes both exploration and exploitation into consideration and tries to find a good balance. However B vs. D usually explores fewer designs

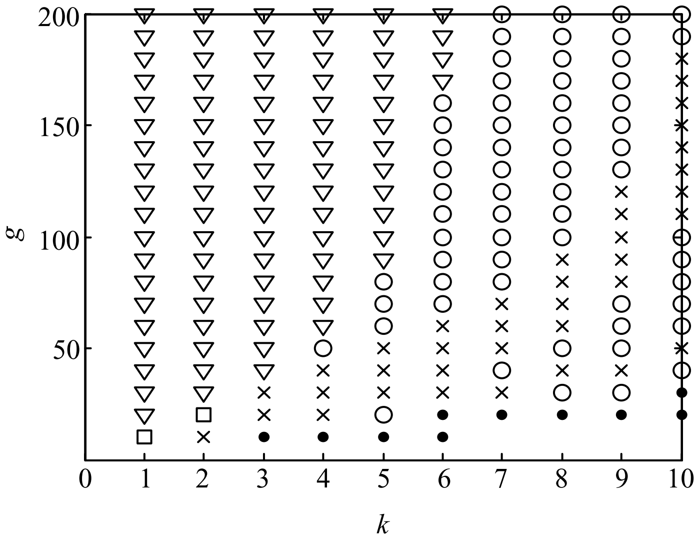


Fig. 3.6. Comparison of HR, OCBA, and B vs. D under large computing budget, middle noise level, and neutral OPC class[12] (Jia et al. 2006a) © 2006 Elsevier

than HR, which does not make any additional effort in exploitation except for equally allocating computing budget units. So it is also reasonable that HR outperform OCBA and B vs. D in these cases.

Third, the crosses appear where the good enough set and the required alignment level are both of middle size/value. Both exploration and exploitation are important, and a good balance is needed. HR pays little attention to do the balance. It is difficult for B vs. D to outperform OCBA with a well-tuned "breadth" (Lin 2000b). This is why OCBA outperforms HR and B vs. D in many of these cases. Since the scenario in Fig. 3.6 has a large computing budget, even in HR each design obtains some exploitation (depth). So HR also performs best in some of these cases (represented by circles).

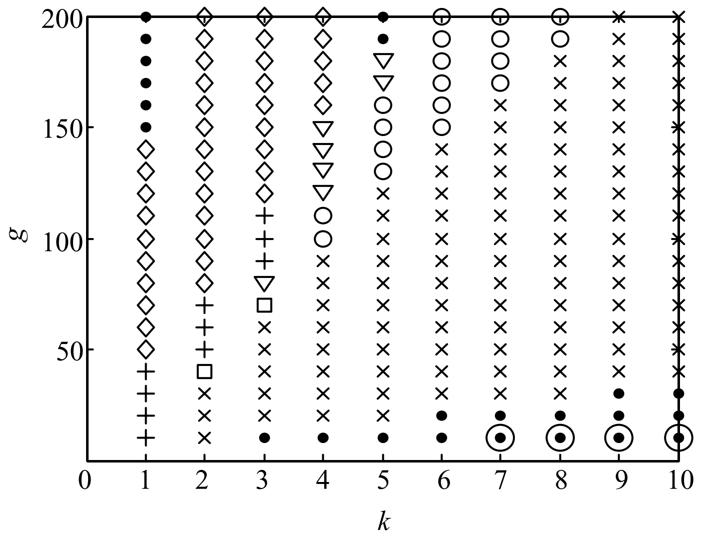
Fourth, the triangles appear where the good enough set is large and the required alignment level is small. These are easy cases and little effort in

[12] "the best" means this selection rule requires the smallest selected subset to achieve the same high AP in the same scenario. Again the experiment mentioned is the one associated with Fig. 3.3 and the text describing it.

balancing exploration and exploitation is needed. So HR, OCBA, and B vs. D require the same subset sizes in these cases.

Following the above analysis, we can summarize a couple of simple rules to choose a good selection rule among the three most frequently used selection rules in OO literature. For instance, in Fig. 3.6, a rule of thumb is: choose HR when the good enough set is small and the required alignment level is high; and choose OCBA in all other cases. These rules are also listed in the end of Section 5.

Now we predict the size of the selected set of these three selection rules using Eq. (3.5) together with the regressed value in Table C.20, C.21, and C.22 in Appendix C. By comparing these predicted values, we predict which selection rule is the best for each (g,k) values, as shown in Fig. 3.7.



● HR is predicted as the best
× OCBA is predicted as the best
+ B vs. D is predicted as the best
○ HR and OCBA are predicted as better
◇ HR and B vs. D are predicted as better
□ OCBA and B vs. D are predicted as better
▽ HR, OCBA, and B vs. D are predicted as equal
⊙ Wrong prediction

Fig. 3.7. Comparison of HR, OCBA, and B vs. D using regressed values under large computing budget, middle noise level, and neutral OPC type (Jia et al. 2006a) © 2006 Elsevier

Note that in all (g,k) combinations, both Fig. 3.6 and Fig. 3.7 recommend a best selection rule. For a (g,k), if the recommended selection rule(s) in Fig. 3.7 is a subset of the ones recommended in Fig. 3.6, we say Fig. 3.7 gives a right recommendation; otherwise, we mark the corresponding case by a big circle. For example, in the bottom right of the figure, some points are marked. The reason is that the corresponding alignment probabilities are smaller than 0.95, as shown in Fig. 3.6. Note that in the aforementioned working range of $20 \le g \le 200$, $s=Z(\bullet/\bullet)<180$, and when the fraction k/g is small, Fig. 3.7 always gives **right** recommendations. This makes an easy way to choose a good selection rule once the computing budget to allocate (T), the noise level (σ), the OPC class, the size of good enough set (g), and the required alignment level (k) are specified. To show how this predication helps to further reduce the size of the selected set, we will use two examples in the next section as demonstration.

3 Examples of search reduction

By using selection rules, OO usually can achieve one order of magnitude or more reduction in search. We demonstrate in this section how to obtain further search reduction by selecting a good selection rule. We use Eq. (3.5) together with the regressed values in Appendix C to predict which selection rule is the best in a given problem. In Section 2, to obtain the regressed values, we assume i.i.d. noises with normal distribution, and explore many combinations of (g,k). However, these results are often universally applicable to problems of a wide range similar to the research in applications of OO previously demonstrated (Ganz and Wang 1994; Xie 1994; Wieselthier et al. 1995; Yang et al. 1997; Cassandras et al. 1998; Guan et al. 2001; Hsieh et al. 2001). We use two examples where either such assumptions are clearly violated or no knowledge concerning assumptions of the problem are available. In the first example, we test a function optimization problem, and show how to use an approximate model to achieve search reduction. In the second example, we test a practical manufacturing queuing network, and use the regressed values to compare HR, OCBA, and B vs. D.

3.1 Example: Picking with an approximate model

We consider again the example used in Section II.6, which shows how to apply our results when the accurate model is deterministic but complex.

We briefly review the problem formulation first. Consider a function defined on the range $\Theta=[0,1]$

$$J(\theta)=a_1\sin(2\pi\rho\theta)+a_2\theta+a_3, \tag{3.6}$$

where $a_1 = 3$, $a_2 = 5$, and $a_3 = 2$. We set $\rho = 500$. So there are five hundred cycles in the range [0, 1]. The true model is deterministic, but rather complex considering relative and absolute minima. To get the exact model of function J and its minima, we need extensive samples of $\theta\in\Theta$, which we pretend to be costly. Instead, we find the trend for J is basically increasing, and then adopt a crude model

$$\hat{J}(\theta)=5\theta. \tag{3.7}$$

This is the linear part of function J. We regard the error between the true and crude models as noise, i.e.,

$$\hat{J}(\theta)=J(\theta)+\text{noise}. \tag{3.8}$$

Assume that we have no prior knowledge on the noise, and that the noise has i.i.d. normal distribution $N(0,\sigma^2)$. By generating 1000 uniform samples of θ from [0, 1]

$$\Theta_N=\{\theta_1,\theta_2,\ldots,\theta_{1000}\},$$

we can estimate the standard deviation σ by

$$\sigma^2=\text{var}_{\theta_i\in\Theta_N}\left(J(\theta_i)-\hat{J}(\theta_i)\right)$$

after adjusting for mean values.

We have totally 1000 computing budget units to allocate, i.e., we can take one observation per design in Θ_N on the average. HR will equally allocate the computing budgets to each design in this way. Other selection rules are different. In the following, we use the regressed values to estimate the minimal subset sizes of different selection rules. Note that the crude model in Eq. (3.7) is linear, so we choose neutral OPC class. After taking one observation of each design θ_i in, we find $\sigma = 0.5$ is a good estimate of the noise level. We try to cover at least k designs in the true top-50 ones in Θ_N, k =1,2,…10, with alignment probability no smaller than 0.95. Using Eq. (3.5) together with the regressed values in Table C.20 in Appendix C, we can estimate the minimal subset size of each selection rule, $\hat{s}$. Note that when evaluating designs, the selection rules use the crude model, and the true top-50 designs are defined by the complex model. To check whether $\hat{s}$ is a good approximate of the true value, we use experiments to test the true value, denoted as s. For a fixed value of s, we take 1000 independent samples of Θ_N to estimate the probability (the ratio of these 1000 experiments) that there are at least k truly top-50 designs in Θ_N contained in the observed top-s designs. Then we regard the minimal value of s to make AP≥0.95 as the true value. We compare the estimated value $\hat{s}$ with the true value s of each selection rule in Table 3.5.

Table 3.5. Estimated subset size of different selection rules in Example 3.1 (Jia et al. 2006a) © 2006 Elsevier

k	HR		OCBA		B vs. D		SPE		HR_gc		HR_ne	
	$\hat{s}$	s	$\hat{s}$	s	$\hat{s}$	s	$\hat{s}$	s	$\hat{s}$	s	$\hat{s}$	s
1	18	19	15	12	14	10	17	12	14	12	13	12
2	28	21	23	18	54	17	28	18	22	17	18	19
3	38	25	32	23	159	29	41	24	31	22	24	23
4	47	32	42	28			54	30	40	27	32	28
5	56	36	52	33			68	37	51	32	41	33
6	65	39	63	38			83	43	61	37	50	38
7	74	44	74	44			98	51	73	41	61	43
8	82	48	86	48			113	58	84	45	72	48
9	90	53	98	52			129	70	96	51	83	52
10	98	56	110	58			145	81	109	56	96	58

For B vs. D when we require to cover more than 4 top-50 designs, both the predicted subset size via the regression model and the true subset size exceed 200. So we do not list those corresponding sizes in Table 3.5. From Table 3.5 we can see that the estimated subset sizes are usually no smaller than the true sizes. This shows the conservative nature of the estimate. When the required alignment level k is specified, by comparing the estimated subset sizes, we can find the best among the selection rules. In Table 3.5, we find that HR_ne has the smallest estimated subset size among all the selection rules of interest for k=1,2,…10. So HR_ne is recommended in these cases. We can also sort the selection rules by the true subset sizes s in Table 3.5, from the smallest to the largest. We show the true order and the true AP of HR_ne among the selection rules for different k in Table 3.6.

Table 3.6. The true order and AP of HR_ne (Jia et al. 2006a) © 2006 Elsevier

k	True order	True AP
1	2	0.961
2	5	0.946
3	2	0.959
4	2	0.976
5	2	0.988
6	2	0.998
7	2	0.999
8	2	1.000
9	2	1.000
10	3	1.000

We compare 6 selection rules in this example. HR_ne is the predicted best one for $k = 1, 2,\ldots,10$. From Table 3.6 we can see that this prediction is good, because HR_ne is within the top-2 selection rules in most k values, with the exception of $k = 2$ and 10. We analyze the two cases with more details. When $k = 2$, HR_ne is the 5th best selection rule. Please note, in that case, the true best subset size is 17, which is achieved by B vs. D and HR_gc. The subset size of HR_ne is 19, which is very close to 17. When k=10, HR_ne is the 3rd best selection rule, with a true subset size of 58. The true best subset size in this case is 56, which is achieved by HR and HR_gc. The two sizes are close to each other. Our method recommends HR_ne, which is a truly good selection rule in this case.

We can also test HR_ne from another aspect. Assume we take the recommendation, choose HR_ne as the selection rule, and use the regressed subset sizes accordingly for each value of k. We use 1000 experiments to estimate the true alignment probability (AP) that the selected subset of designs can cover at least k top-50 designs. The results are also shown in Table 3.6. From Table 3.6, we can see that the alignment probabilities are higher than the required value, 0.95, in most cases. There is only one exception, $k = 2$, in which AP is 0.946, very close to the required value 0.95. This example shows that the regression function can give conservative estimate of subset size and our method performs well in comparing selection rules. This is useful in practice. First we use the regressed value to estimate the subset sizes of different selection rules and find the best one. Using this best rule, we can obtain a subset of designs, which contains at least k true good enough designs with high probability. This often leads to a search reduction of at least one order of magnitude. As shown in this example, when the true model is complex, we can use a crude one in subset selections. Based on the crude model, using our method, we can recommend HR_ne as a good selection rule, which often is a truly good selection rule.

Exercise 3.3: Using the above method, we can easily find a good selection rule for a given problem. After we determine which selection rule to use, we can use the regressed values in Appendix C to calculate the size of the selected set for that selection rule. In this example, the estimated sizes of the selected set of HR_ne are shown in Table 3.5. However, when we take a close look at Table 3.5, we find the estimate $\hat{s}$ sometimes is too conservative from the true value s. How can we improve this estimate? [Hint: Section VIII.3 introduces a way to improve this estimate.]

3.2 Example: A buffer resource allocation problem

We consider a 10-node network as shown in Fig. 3.8. Such a network could be the model for a large number of real-world systems, such as a manufacturing system, a communication network, or a traffic network. For details about the background of this example, please refer to (Chen and Ho 1995; Patsis et al. 1997). We will test whether our method can recommend good selection rules on this practical model. There are two classes of customers with different inter-arrival distributions but having the same service requirement. We use c1 and c2 to denote the two classes of of customers. Their inter-arrival times are with uniform and exponential distributions respectively, i.e., c1: *U*[2, 18], c2: *Exp*(0.12). The 10-node network represents a connected three service stages. Node 0-3 denotes the first stage,

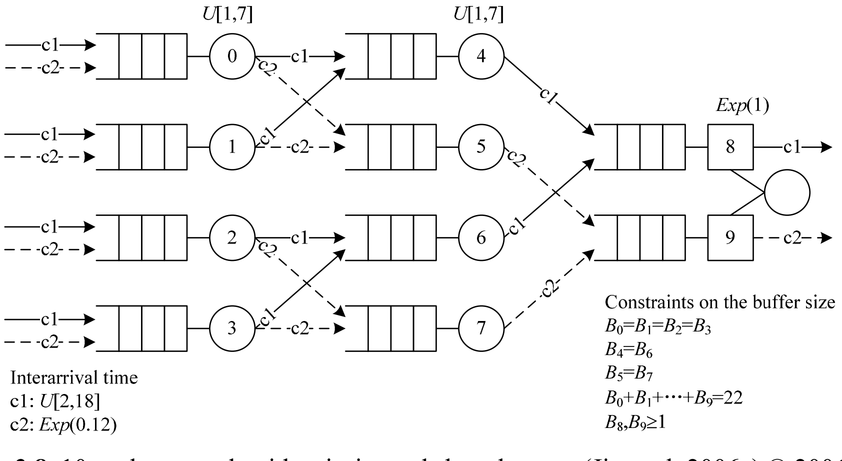


Fig. 3.8. 10-node network with priority and shared server (Jia et al. 2006a) © 2006 Elsevier

where customers enter the system. Node 4-7 denotes the second stage, where customers from different classes are served separately. Node 8-9 denotes the last stage, where customers leave the system after the service completes. The service time is with uniform distribution at node 0-7, i.e., $U[1,7]$, and is with exponential distribution with rate 1.0 at node 8 and 9, i.e., $Exp(1)$. All the nodes from 0 to 7 have their own servers, while node 8 and 9 share one server. Each node has a buffer with size B_i, $I = 0,1,\ldots 9$, (not including the one being served). It is the allocation of buffer size that makes this problem interesting. A buffer is said to be full if the queue length (not including the one being served) equals to the buffer size. If the buffer in the downstream is full when one node finishes serving one customer, this customer cannot leave and this server is idle and blocked. When two nodes are blocked by the same node in downstream, "first blocked, first served" is applied.

The two classes of customers have different priorities.

1. In queue of node 0-3, c1 customers jump before c2 customers. If a c1 customer arrives when a c2 customer is right in the process of being served, this service is allowed to be completed.
2. At node 8, when the queue length is greater than one, a c1 customer can be served. If a c2 from node 9 is in the process of being served while the queue length at node 8 becomes greater than one, the service to c2 will be interrupted to allow high priority (c1 customer).

For symmetric reason, we set the following constraints on buffer sizes:

$$B_0=B_1=B_2=B_3$$
$$B_4=B_6$$
$$B_5=B_7$$
$$B_8\geq 1,\ B_9\geq 1.$$

We consider the problem of allocating 22 buffer units. There are totally 1001 different configurations. We want to find one configuration that can minimize the expected processing time for the first 100 customers, assuming there are no customers in the network initially.

For each of the 1001 configurations, we do 200 experiments and use the mean value to estimate the processing time for the first 100 customers. Although 200 experiments are not enough to discriminate the performance of different configurations exactly, the estimated top-100 designs are much more accurate than based on only one observation per design. We assume that there are totally 1001 computing budget units to allocate in the following experiments, as what we have mentioned above. So we sort the configurations using the estimated value and regard this as the true order.

We want to find at least k of top-g designs (e.g., $g = 100$, $k = 1,2,\ldots 10$) with high probability (e.g., $\alpha \geq 0.95$). Because HR, OCBA, and B vs. D are frequently used in ordinal optimization, we focus on comparing the three selection rules in this example. Since we have little knowledge of the performance of the network, we use neutral OPC class to approximate the true OPC of this system. From preliminary experiments, we find that 0.5 is a good estimate of noise level after adjusting for mean values. Assume that we have totally 1001 observations. Using the regressed values, we can estimate the subset sizes for each selection rule, $\hat{s}$. For each selection rule, we use 1000 experiments to estimate the true subset sizes, s, as we did in Section 2. We show the two groups of subset sizes in Table 3.7.

Table 3.7. Estimated and true subset sizes for Example 3.2 (Jia et al. 2006a) © 2006 Elsevier

k	HR		OCBA		B vs. D	
	$\hat{s}$	s	$\hat{s}$	s	$\hat{s}$	s
1	11	7	11	2	9	1
2	16	11	14	3	18	3
3	21	14	17	5	41	16
4	25	18	20	7		
5	30	22	24	10		
6	34	25	28	13		
7	38	28	32	16		
8	42	32	36	19		
9	46	35	40	23		
10	50	39	45	26		

When we use B vs. D to cover at least $k(k \geq 4)$ top-100 designs with a probability no smaller than 0.95, the true subset sizes(s) should be no greater than 200. So we do not list the values in Table 3.7. Recall also that the working range of the regression function in this case ($T = 1000$, middle noise level and B vs. D) is $1 \leq k \leq 3$ and $k/g \leq 1/25$. So we only show the predicted subset sizes $\hat{s}$ and the true sizes s when $k = 1$, 2, and 3 for B vs. D. We also use 1000 replications to test the alignment probability for the estimated subset sizes of different selection rules, which we show in Table 3.8, regard as the true APs.

In Table 3.7, the estimated subset sizes $\hat{s}$ are usually greater than the true values s. In Table 3.8, the true alignment probabilities are always no smaller than the required value 0.95. This indicates that the estimated subset sizes are conservative and close to true values. We also list the predicted and truly best selection rule(s) under different *k* values, as is shown in Table 3.9.

Table 3.8. True alignment probability when using estimated subset size (Jia et al. 2006a) © 2006 Elsevier

k	HR	OCBA	B vs. D
1	0.994	1.000	0.998
2	0.999	1.000	0.993
3	0.999	1.000	0.975
4	0.999	1.000	
5	0.997	1.000	
6	0.999	0.999	
7	0.999	0.999	
8	0.999	1.000	
9	0.999	0.999	
10	0.999	1.000	

Table 3.9. True and predicted best selection rules for different *k* (Jia et al. 2006a) © 2006 Elsevier

k	Estimated best	Truly best
1	B vs. D	B vs. D
2	OCBA	OCBA and B vs. D
3	OCBA	OCBA
4	OCBA	OCBA
5	OCBA	OCBA
6	OCBA	OCBA
7	OCBA	OCBA
8	OCBA	OCBA
9	OCBA	OCBA
10	OCBA	OCBA

In Table 3.9, the predicted best selection rules are always a subset of the truly best ones. Thus, we can use the regressed value to predict what the best selection rule is. The recommended selection rule is truly the best.

Although our regressed values are obtained under the assumption of i.i.d. noise with normal distribution, it works well in other settings. For example, in Example 3.1 we regard the deterministic but complex error between the true model and the crude mode as observation noise. In Example 3.2, we do not know the true OPC type of the system and use neutral class as an approximate. From more experiments, we find the observation noise for different designs are not independently and identically distributed. However, numerical results have shown that our prediction works well (Jia et al. 2004).

Exercise 3.4: Are the symmetric constraints on the buffer size reasonable? If we remove the symmetric constraints on the buffer size, can we find a better solution?

4 Some properties of good selection rules

In Section 2 and 3, we have developed a method to predict the best selection rule among the nine considered in this chapter. However, as mentioned at the beginning of this chapter, there are a huge number of selection rules. The ones we discussed so far are only a small portion of all the possible selection rules. Understanding the method presented in the previous two sections can suffice the current practical application. It may also interest some readers if we can discover some general properties of good selection rules. These rules can then serve as a guideline for us to look for better selection rules in the future. In this section, we discuss three such properties, namely, ***1) without elimination, 2) global comparison, and 3) using mean value of observations as the estimate of design performance.*** (The three properties will be explained in details later in this section.) We use experiments to show that a selection rule with property 1) is no worse than others and actually is strictly better in 81% of all the tested scenarios; a selection rule with property 2) is no worse than others and actually is strictly better in 71% of all the tested scenarios; a selection rule with property 3) is no worse than others and actually is strictly better in 39% of all the tested scenarios. For those who want to repeat these numerical experiments, the detailed experimental data can be found in Section 4.4 in (Jia 2006).

Let's now consider the first property: without elimination. If a selection rule uses elimination, then a design that fails in an early round will have no chance to receive further observation, and will finally receive a low observed order. Sequential pair-wise elimination (SPE) is one of the selection rules that use elimination. HR_ne is not. To see how this property affects the performance of a selection rule, we compare SPE and HR_ne in all the five OPC types and three noise levels. Let there be a large computing budget. For $g = 20$, $s = 10, 20, \ldots, 200$ and different values of k, we use 1000 independent replications to test the AP in each case. The s value such that the corresponding AP is no less than 0.95 is the minimal size of the selected subset of that selection rule in that scenario. By comparing the sizes of the selected subset of SPE and HR_ne in all the scenarios mentioned above, we find HR_ne does not require a larger selected set than SPE in all the tested scenarios, and in 81% of the scenarios requires a smaller subset.

This justifies that without elimination is a property for good selection rules.

Then let's consider the second property: global comparison. A selection rule with global comparison will compare all the designs that enter a round together. A selection rule without global comparison will only compare some of the designs that enter a round. SPE uses pair-wise comparison, thus is an example of without global comparison. HR_gc compares all the designs in a round together, thus is a selection rule with global comparison. Since HR_gc and SPE both use elimination, by comparing these two rules, we can see how the global comparison affects the performance of a selection rule. Note that the Basic Property in Section 1 tells us that it does not hurt if we select the observed top-s designs. In each round HR_gc selects the observed top-half designs. This is consistent with the Basic Property. So, in principle, HR_gc should not be worse than SPE in all the scenarios. The question is: Since SPE is not consistent with the Basic Property, how much worse is this selection rule than HR_gc? We compare SPE and HR_gc in all the five OPC classes and three noise levels. Other parameter settings are also the same as when we used to compare SPE and HR_ne to test the first property. The result is: HR_gc is no worse than SPE in all the tested scenarios, and is better in 71% of the scenarios. This justifies that global comparison is a property of good selection rules.

Finally, we will consider the third property: using mean value of observations as the estimate of design performance. In some selection rules, the designs are ordered according to the number of "wins" they receive. RR, SPE, HR_gc are some of the examples. Some other selection rules order the designs according to the mean value of observations, e.g., HR and HR_CRR. To see how this property affects the performance of a selection rule, we compare HR_CRR and RR. Both selection rules equally assign the computing budget among the designs. Note, according to the Basic Property, selecting the observed top-s designs does not hurt in this case, which is exactly what HR_CRR does. So, in principle HR_CRR is no worse than RR. The question is: How much worse can RR be than HR_CRR? We compare HR_CRR and RR in all five OPC types and three noise levels. Let g = 50. There are large computing budgets. The comparison shows that HR_CRR is no worse than RR in all the tested scenarios, and is better in 39% of the scenarios. This justifies that using the mean value of observations as the estimate of design performance is a property of good selection rules.

5 Conclusion

We use regression functions to quantify the efficiency of different selection rules for ordinal optimization, especially HR, OCBA, and B vs. D, which are frequently used. Using the regressed values, we can predict the best selection rule(s) of interest under different parameter settings. The prediction is rather good, as we have shown in section 3 by numerical examples. We showed that some selection rules are no worse than some others, and discussed the properties of good selection rules from three aspects: Without elimination, global comparison, and using mean value as the measure.

To predict the best selection rule among a given set of selection rules using our method, we summarize the application procedure as follows (Box 3.1).

Box 3.1. How to predict a good selection rule

Step 1:	Estimate the OPC class of the system and the noise level in observation.
Step 2:	Specify the values of k, g, and α. We want to cover at least k designs in true top-g ones with high probability no smaller than α.
Step 3:	Specify how much computing budget you have, i.e., you can simulate the system how many times during the entire optimization.
Step 4:	Steps 1-3 determines a specific scenario. Find the regressed values of the coefficients in Eq. (3.5) in Appendix C for this scenario.
Step 5:	Use Eq. (3.5) to predict the size of the selected subset for each selection rule. By comparing the predicted subset sizes, we can easily predict which selection rule is the best.
Step 6:	Using the recommended selection rule, we can obtain a subset of designs, which contains at least k good enough designs with high probability.
Step 7:	In this way, we can usually have a search reduction of at least one order of magnitude.

Based on ample experimental data, we also summarize a couple of simple, quick and dirty tips for easily picking up a good selection rule without calculation steps above. They are as follows:

1. In most of the cases, we recommend HR_ne, which works well and is a good selection rule among all the 9 selection rules compared in this paper.

2. In extremely difficult case (the computing budget is small, the size of good enough subset is small, and we try to cover many good enough designs), we recommend HR.

The above tips have intuitive illustrations. HR_ne has all the three properties of optimal selection rules mentioned in Section 4, which are without elimination, global comparison, and using mean value as the measure. Numerical results show each property can improve the alignment probability of selection rule separately. So HR_ne is a good choice in most cases. However, when the computing budget is extremely small and we can only explore part of the designs, exploration is more important than exploitation. Since HR usually explores a larger number of designs than HR_ne, it is a good choice in the latter difficult cases.

Since HR, OCBA, and B vs. D are the three frequently used rules in ordinal optimization, we also summarize some simple rules to choose a good one among them:

1. When the computing budget is small and we do not want much calculation to allocate the computing budget, HR is a good choice.
2. For special sizes of middle and large computing budget (i.e., there are 1000 designs and we have 1000 or 30000 computing budget units to allocate), OCBA is a good choice. Especially we prefer the following parameter settings in OCBA. When $T = 1000$, we prefer $m_0 = 500$. When $T = 30000$, $m_0 = 1000$.
3. In all other cases where we need a good and automatic balance between exploration and exploitation, B vs. D is a good choice.

Note that OCBA fixes the "breadth" and B vs. D can increase the "breadth" during the allocation procedure. So OCBA with an optimal "breadth" may have the same high alignment probability as B vs. D (Lin 2000b). This is why we prefer OCBA in the second rule above. We have used ample experiments to find the optimal "breadth" of OCBA in those cases. However, as pointed out in the third rule, in general cases we prefer B vs. D, which can do the balance automatically.

Finally and more generally, we can regard the selection rule as a way of narrowing down the search for a good enough solution. Currently, the practice of OO is first to narrow down from Θ to Θ_N. Denote this as the first stage. It typically uses uniform sampling to get a representative set from Θ. Then in the second stage we narrow down from Θ_N to S. This process is what we do with the selection rules, as has discussed in this chapter. However, this is only one possible way of doing the search for the "good enough". There still exist a number of other possibilities. For example, sampling in the first stage can be enhanced. Instead of uniform sampling, we can use heuristics to bias sampling towards more good designs. There need not be only two stages, but many stages leading to what may be called iterative OO (Deng and Ho 1997). In short, we can view each stage of the selection as overlaying a selection probability density over the set of design possibilities in question. Good selection rules are the ones that favor the good designs. More discussion on this will be found in Chapter VII.