Information Utilization Ratio in Heuristic Optimization Algorithms

Junzhi Li and Ying Tan
Computational Intelligence Laboratory
Peking University
{ljjz,ytan}@pku.edu.cn

Abstract
Heuristic algorithms are able to optimize objective functions efficiently because they use intelligently the information of the objective functions. Thus information utilization is vital to the performance of heuristics. However, the concept of information utilization has remained vague and abstract because there is no reliable metric to reflect the extent to which the information of the objective function is utilized in heuristic algorithms. In this paper, the metric of information utilization ratio (IUR) is defined, which is the ratio between the utilized information quantity and the acquired information quantity in the searching process. IUR proves to be well-defined. Several examples of typical heuristic algorithms are given to demonstrate the procedure of calculating IUR. The results also reveal that elevating IUR is the potential cause of many algorithmic improving works. IUR can be an index of how exquisite an algorithm is designed and guide the design of new heuristics and the improvement of existing ones.

1 Introduction
In the field of computer science, many heuristic algorithms have been designed to solve complex optimization problems. Although optimal solutions are not guaranteed to be found, heuristics can often find acceptable solutions with reasonable cost.

The no free lunch theorem [Wolpert and Macready, 1997] has revealed that no algorithm performs better than random search when no prior distribution of the objective function is given. However, in real world applications, we are usually interested in a certain kind (or distribution) of objective functions, where the heuristic algorithms are useful. In this case, there is free lunch, because the objective function (and resultantly its optimal point) can be identified with limited information (which depends on the entropy of the distribution). For example, only three times of evaluation are needed to locate the center of a two-dimensional sphere function [Auger and Teytaud, 2010]. Assume the information acquired in each evaluation is fixed, then how much information is utilized by the algorithm per each evaluation determines the lower bound of required evaluation number to locate the optimal point. In other words, the extent of information utilization determines the upper bound of an algorithm’s performance considering a certain kind of objective functions. While the actual performance also depends on the way of information utilization. If the way accord with (the kind of) the objective function, then it performs well. So, it is a common sense that the way and the extent of information utilization in a heuristic algorithm are both vital to its performance.

So far, many works have compared the ways of information utilization in heuristic algorithms, but there is no reliable metric to compare the extents of information utilization. The key to distinguish extents of information utilization is the following insight: most algorithms actually don’t require (use) the exact evaluation values of all these evaluated solutions. For example if the algorithm compare two evaluation values to decide which one is better, the expected utilized (or required) information is actually no more than one bit.

In this paper, based on some basic concepts of the information theory, a formal definition of information utilization ratio (IUR) is proposed to reflect to what extent the information of the objective function is used by a heuristic algorithm, which is the ratio between the information quantity that is utilized and the information quantity that is acquired in the searching process. It proves to be well-defined.

The IURs of several typical heuristic algorithms are given as examples to demonstrate the procedure of calculating this metric. It’s not difficult to calculate IURs for most heuristic algorithms. At least some lower and upper bounds can be given with limited efforts. Through these examples, we also show the tendency that for each family of heuristics (where they share similar ways of information utilization), usually IUR gets higher and higher with the development of these algorithms.

IUR is a useful index for guiding the design of new heuristics and the improvement of existing algorithms because it reflects how exquisite an algorithm is designed and determines the upper bound of the algorithm’s performance.

The remainder of this paper is organized as follows. The basic definitions and some useful propositions are given in Section 2. The IURs of several heuristic optimization algorithms are calculated as examples in Section 3. Some related issues are discussed in Section 4. Finally Section 5 concludes
2 Definitions

The definitions and basic properties of information entropy $H(.)$, joint entropy $H(.,.)$ and conditional entropy $H(.,.)$ are frequently used in this paper, which however cannot be present here due to limitation of space. We suggest readers unfamiliar with information theory refer to the original paper [Shannon, 1948] or any tutorial.

The following lemma defines a useful function for calculating IURs of many algorithms.

Lemma 1. If $\eta_1, \eta_2, \ldots, \eta_{g+1} \in \mathbb{R}$ are independent identically distributed random variables,

$$H(1(\min(\eta_1, \eta_2, \ldots, \eta_g) < \eta_{g+1})) = -\frac{g}{g+1} \log \frac{g}{g+1} - \frac{1}{g+1} \log \frac{1}{g+1} \triangleq \pi(g),$$

where

$$I(x < y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$$

is the indicator function.

Definition 1 (Objective Function). The objective function is a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ is a totally ordered finite set with $|\mathcal{Y}| = n$ and for any $x \in \mathcal{X}$, $f(x) \in \mathcal{Y}$ is independently and uniformly distributed.

For any $x \in \mathcal{X}$, $H(f(x)|x) = \log n$.

For the convenience of further discussion, the range of the objective function is assumed to be a finite set. Nothing else is known about the objective function. It is totally a black box, so there is definitely no free lunch for any algorithm. Thus, how to design a better algorithm cannot be discussed under this assumption. But for discussing the ratios of information utilization in heuristic optimization algorithms, this is enough. Of course readers can use practical prior distributions instead.

Definition 2 (Optimization Algorithm). An optimization algorithm is defined as follows.

1: $i \leftarrow 0$.
2: $D_0 \leftarrow \emptyset$.
3: repeat
4: $i \leftarrow i + 1$.
5: Sample $X_i \in 2^\mathcal{X}$ with distribution $\mathcal{A}_i(D_{i-1})$.
6: Evaluate $f(X_i) = \{f(x)|x \in X_i\}$.
7: $D_i \leftarrow D_{i-1} \cup \bigcup_{x \in X_i} \{x, f(x)\}$.
8: until $i = g$.

Where $\mathcal{A}_i$ is a mapping from $2^\mathcal{X} \times \mathcal{Y}$ to the set of all distributions over $2^\mathcal{X}$, and $g$ is the maximal iteration number.

In each iteration, the input of the algorithm is the history information, which is a subset of $\mathcal{X} \times \mathcal{Y}$, and the output is a distribution over $2^\mathcal{X}$, with which the solutions to be evaluated next are drawn. Note that the output $\mathcal{A}_i(D_{i-1})$ is deterministic given $D_{i-1}$.

Definition 3 (Information Utilization Ratio). If $\mathcal{A}$ is an optimization algorithm, the information utilization ratio of $\mathcal{A}$ is defined as

$$IUR_\mathcal{A}(g) = \frac{H(X, Z) - \sum_{i=1}^{g} H(X_i|Z_i)}{\sum_{i=1}^{g} H(Y_i|X_i)},$$

where $g$ is the maximal iteration number,

$X = \{X_1, X_2, \ldots, X_g\}$

is the set of all sets of evaluated solutions,

$Y = \{f(X_1), f(X_2), \ldots, f(X_g)\}$

is the set of all sets of evaluation values,

$Z = \{\mathcal{A}_1(D_0), \mathcal{A}_2(D_1), \ldots, \mathcal{A}_g(D_{g-1})\}$

is the output distributions in all iterations of algorithm $\mathcal{A}$.

Theorem 1. If $0 < \frac{g}{\sum_{i=1}^{g} H(Y_i|X_i)} < \infty$, then $IUR_\mathcal{A}(g)$ exists and $0 \leq IUR_\mathcal{A}(g) \leq 1$.

Proof. Let $X_i \triangleq \{X_1, \ldots, X_i\}$, $Y_i \triangleq \{Y_1, \ldots, Y_i\}$, $Z_i \triangleq \{Z_1, \ldots, Z_i\}$.

$$H(X, Z) - \sum_{i=1}^{g} H(X_i|Z_i) = \sum_{i=1}^{g} H(X_i, Z_i|X_{i-1}, Z_{i-1}) - \sum_{i=1}^{g} H(X_i|X_{i-1}, Z_i) = \sum_{i=1}^{g} H(X_i|X_{i-1}, Z_i) - \sum_{i=1}^{g} H(Y_i|X_i)$$

$$= \sum_{i=1}^{g} H(Z_i|X_{i-1}) - \sum_{i=2}^{g} H(Z_{i-1}|X_{i-1}) = \sum_{i=2}^{g} H(Z_{i-1}|X_{i-1}) - \sum_{i=2}^{g} H(Y_{i-1}|X_{i-1})$$

$$= \sum_{i=2}^{g} H(Z_{i-1}|X_{i-1}) + \sum_{i=2}^{g} H(Y_{i-2}|X_{i-1}) + \sum_{i=2}^{g} H(Y_{i-1}|X_{i-1}) - \sum_{i=1}^{g} H(Y_i|X_i)$$

$$= \sum_{i=2}^{g} H(Y_{i-1}|X_{i-1}) + \sum_{i=1}^{g} H(Y_{i-1}|X_i) = \sum_{i=2}^{g} H(Y_{i-1}|X_{i-1}) + \sum_{i=1}^{g} H(Y_{i-1}|X_i)$$

$$\leq \sum_{i=1}^{g} H(Y_i|X_i).$$
The lower and upper bounds are both tight.

The numerator in the definition of IUR is quite obscure at first glance. Actually it means the quantity of the information of the objective function which is utilized (or required) by the algorithm. Firstly, the uncertainty of $X$ and $Z$ only lies in two aspects: the random sampling step and the lack of the information from $Y$. Thus the latter part can be seen as the objective function’s information that is utilized by the algorithm. Secondly,

$$\sum_{i=1}^{g} H(Z_i|X_{i-1}, Z_{i-1}) = \sum_{i=1}^{g} H(Z_i|X_{i-1}, Z_{i-1}) - \sum_{i=1}^{g} H(Z_i|X_{i-1}, Z_{i-1}, Y_{i-1})$$

is similar to the concept of information gain in classification problems, which also indicates the contribution of the information of $Y$ to the algorithm. Thirdly, $H(Y_{g-1}|Z, X_{g-1}) + H(Y_{g}|X_{g})$ can be seen as the wasted information of $Y$. Because 1) the evaluation values in the last iteration cannot be utilized and 2) the information of $Y$ is fully utilized only if $Y_{g-1}$ can be reconstructed with $Z$ given $X_{g-1}$. While the denominator is obviously the objective function’s information that is acquired by the algorithm.

It’s very important to note that only the information about the objective function is considered in this definition. The use of other information, like positions of the individuals in the algorithm, is irrelevant here. Algorithms could update these positions even without any evaluation, which would be meaningless. Since the task is to find the optimal point(s) of the objective function, the usage of the objective function’s information is the most important.

In most cases, it is unwise to calculate IUR by definition. Although to calculate the denominator is quite straightforward under the independent uniform assumption, which equals the number of evaluations times log $n$, to calculate the joint entropy $H(X, Z)$ is difficult and unnecessary. In each iteration, the output $σ(D_{i-1})$ is a certain distribution, which is usually determined by some parameters in the algorithm. In fact we can certainly find (or construct) the set of intermediate parameters $M_i$, such that there is a bijection from $M_i$ to $Z_i$ given $X_{i-1}$ and $M_i$ is determined only by $Y_{i-1}$ (otherwise $H(Z_i|X_{i-1}, Y_{i-1}) > 0$), then

$$\sum_{i=2}^{g} H(Z_i|X_{i-1}, Z_{i-1}) = \sum_{i=2}^{g} H(M_i|M_{i-1}) = H(M).$$

We only need to know the information quantity that is required to determine these intermediate parameters.

3 Information Utilization Ratios of Typical Optimization Algorithms

3.1 Random Search Algorithms

Random Search

Random search (RS) is often considered as a baseline for optimization algorithms. It is not a heuristic algorithm and usually fails to find acceptable solutions. If the maximal evaluation number is $m$, RS just uniformly randomly choose $m$ solutions from $X$.

RS doesn’t utilize any information of the objective function because $Z$ is fixed.

Proposition 1.

$$IUR_{RS} = 0.$$ (17)

Luus-Jaakola

Luus-Jaakola (LJ) [Luus and Jaakola, 1973] is a heuristic algorithm based on random search. In each iteration, the algorithm generates a new individual $y$ with uniform distribution within a hypercube whose center is the position of the current individual $x$. If $f(y) < f(x)$, $x$ is replaced by $y$; otherwise, the radius of the hypercube is multiplied by 0.95.

The output of LJ in each iteration is the uniform distribution within the hypercube, which is determined by the position $x$ and the radius. They are both controlled by the comparison result, i.e., $I(f(y) < f(x))$. $f(y)$ is uniformly distributed, but $f(x)$ is the best in the history. Thus,

$$H(M_i|M_{i-1}) = H(I(f(y) < f(x))|M_{i-1}) = \pi(i-1).$$

Proposition 2.

$$IUR_{LJ}(g) = \frac{\sum_{i=1}^{g-1} \pi(i)}{g \log n}.$$ (18)

For example, if $n = 1024$, $g = 10$, then $IUR_{LJ} = 0.0621$, which is independent of the base of the logarithm (similarly hereinafter).

3.2 Evolution Strategies

$$(\mu, \lambda)$-Evolution Strategy

$(\mu, \lambda)$-evolution strategy (ES) [Bäck et al., 1991] is an important heuristic algorithm in the family of evolution strategies. In each generation, $\lambda$ new offspring are generated from $\mu$ parents by crossover and mutation with normal distribution, and then the parents of a new generation are selected from these $\lambda$ offspring. As a self-adaptive algorithm, the step size for mutation is itself mutated along with the position of an individual.

The distribution for generating new offspring is determined by the indexes of the best $\mu$ of the $\lambda$ individuals. Each set of $\mu$ candidates has the same probability to be selected. $H(M_i|M_{i-1}) = H(M_i) = \log (\binom{\lambda}{\mu})$, where $\binom{\lambda}{\mu} = \frac{\lambda!}{\mu!(\lambda-\mu)!}$.

Proposition 3.

$$IUR_{(\mu, \lambda)-ES}(g) = \frac{(g-1) \log (\binom{\lambda}{\mu})}{g\lambda \log n}.$$ (19)

For example, if $n = 1024$, $g = 10$, $\lambda = 10$, $\mu = 5$, then $IUR_{(\mu, \lambda)-ES} = 0.0718$.

Covariance Matrix Adaptation Evolution Strategy

In order to more adaptively control the mutation parameters in $(\mu, \lambda)$-ES, a covariance matrix adaptation evolution strategy (CMA-ES) was proposed [Hansen and Ostermeier, 1996]. CMA-ES is a very complicated algorithm, which adopts several different mechanisms to adapt the mean, covariance matrix and step size for the mutation operation. It is very
efficient on benchmark functions \cite{Auger_and_Hansen_2005,Loshchilov_2013} especially when restart mechanisms are adopted, which won the first place in CEC05 \cite{Hansen_2006} and CEC13 \cite{Loshchilov_2013} competitions. CMA-ES cannot be introduced here in detail, we suggest interested readers refer to an elementary tutorial: \cite{Hansen_2005}.

Given $\mu_{n-1}$, the mean, covariance matrix and step size of the distribution is determined by the indexes and the rankings of the best $\mu$ individuals in each iteration in history. $H(M_n) = \log \frac{\lambda^\mu}{(\lambda-\mu)^n}$.

**Proposition 4.**

\[ IUR_{CMA-ES}(g) = \frac{(g-1) \log \frac{\lambda^\mu}{(\lambda-\mu)^n}}{g \lambda \log n}, \]  

(20)

For example, if $n = 1024, g = 10, \lambda = 10, \mu = 5$, then $IUR_{CMA-ES} = 0.1340$.

It is obvious that $IUR_{CMA-ES} > IUR(\mu, \lambda, ES)$, because not only the indexes of the $\mu$ best individuals, but also their rankings are used in CMA-ES (to calculate their weights, for example).

### 3.3 Particle Swarm Algorithms

#### Particle Swarm Optimization

Particle swarm optimization (PSO) \cite{Eberhart_and_Kennedy_1995} is one of the most famous swarm and heuristic algorithms which is quite simple but surprisingly efficient in numerical optimization. In PSO, a fixed number ($s$) of particles moves in the search space to find the optimal solutions. The position of a particle is updated as follows. In generation $g$, for each particle $i$ and each dimension $j$,

\begin{align*}
    v_{ij}(g + 1) & = v_{ij}(g) + \phi_1 r_{1ij}(pbest_{ij}(g) - x_{ij}(g)) \\
    & + \phi_2 r_{2ij}(gbest_{j}(g) - x_{ij}(g)), \tag{21}
    \\
    x_{ij}(g + 1) & = x_{ij}(g) + v_{ij}(g + 1), \tag{22}
\end{align*}

where $\phi_1$ and $\phi_2$ are constant coefficients, $r_1$ and $r_2$ are random numbers, $pbest$ is the best position in history found by this particle and $gbest$ is the best position found by the entire swarm.

The output distribution in each generation is determined by $I(f(x_i(g))) < f(pbest_t(g - 1))$ and $\arg \min f(pbest_t(g))$.

Although it’s difficult to calculate $H(M)$, we have the lower and upper bounds:

\[ s \sum_{i=1}^{g-1} \pi(i) \leq H(M) \leq \sum_{i=2}^{g} H(M_i) \leq (g-1) \log s + s \sum_{i=1}^{g-1} \pi(i). \]  

(23)

**Proposition 5.**

\[ s \sum_{i=1}^{g-1} \pi(i) \leq IUR_{PSO}(g) \leq \frac{(g-1) \log s + s \sum_{i=1}^{g-1} \pi(i)}{sg \log n}. \]  

(24)

For example, if $n = 1024, s = 10, g = 10$, then $0.0621 \leq IUR_{PSO} \leq 0.0920$.

#### Standard Particle Swarm Optimization

After years of development, many improvements and variants are proposed for PSO. In order to construct a common ground for further researches, a standard particle swarm optimization (SPSO) was defined \cite{Bratton_and_Kennedy_2007}. Compared with original PSO, there are two main modifications: the local ring topology and the constricted update rule. The constricted update rule uses a new coefficient derived from $\phi_1$ and $\phi_2$ to constrict the velocity to guarantee convergence. In the local ring topology, the $gbest$ in the velocity update equation is replaced with a $lbest$, which is the best position among this individual and its two neighbourhoods on the ring.

For each group (consisting of three particles), information with quantity at most $\log 3$ is needed to decide $lbest$.

**Proposition 6.**

\[ s \sum_{i=1}^{g-1} \pi(i) \leq IUR_{PSO}(g) \leq \frac{s(g-1) \log s + s \sum_{i=1}^{g-1} \pi(i)}{sg \log n}. \]  

(25)

For example, if $n = 1024, s = 10, g = 10$, then $0.0621 \leq IUR_{PSO} \leq 0.2047$.

In fact $IUR_{PSO} < IUR_{SPSO}$ though their exact values are difficult to derive. It turns out that the information utilization ratio of the local model is larger than the global model because in local topology the particles interact with each other more frequently.

### 3.4 Differential Evolution Algorithms

#### Differential Evolution

Differential evolution (DE) \cite{Storm_and_Price_1997} is a powerful heuristic algorithm for numerical optimization. The number of individuals in DE is also fixed. The mutation is conducted as below (take DE/rand/1 as an example). For each $x$ in the population, generate

\[ z = x_{r1} + F(x_{r2} - x_{r3}), \]  

(26)

where $r1, r2$ and $r3$ are random indexes and $F$ is a constant coefficient. Then a crossover is conducted between $z$ and $x$ to generate a new candidate $y$, where there is a parameter $CR$ to control the probability that a dimension of $y$ is identical to that of $z$. If $f(y) < f(x)$, $x$ is replaced with $y$, otherwise, $x$ is kept.

In DE, the distribution of generating new offspring is determined by $I(f(y) < f(x))$ of each individual. So the IUR of DE is equal to that of LJ with the same $g$. However, they would be different with the same number of evaluation times.

**Proposition 7.**

\[ IUR_{DE}(g) = \frac{s \sum_{i=1}^{g-1} \pi(i)}{sg \log n}. \]  

(27)

For example, if $n = 1024, s = 10, g = 10$, then $IUR_{DE} = 0.0621$.

IURs of some other DE variants are given in Table 1.
Table 1: IURs of other DE variants

| Algorithm               | IUR          |
|-------------------------|--------------|
| DE/best/1               | IURPSO       |
| DE/current-to-best/1    | IURPSO       |
| DE/rand/2               | IURDE        |
| DE/best/2               | IURPSO       |

**JADE**

JADE [Zhang and Sanderson, 2009] is an important development of DE. There are three main adaptations proposed in JADE:

1. A DE/current-to-pbest/1 mutation strategy. In JADE, 
   
   \[ z_i = x_i + F_i(x_{\text{best}}^p - x_i) + F_i(x_{r1} - x_{r2}) , \]

   where \( x_{\text{best}}^p \) is a randomly chosen individual from the 100p\% best individuals.

2. An optional external archive.

3. Adaptive mutation parameters.

External archive is a useful tool to improve information utilization. However, in JADE these individuals are just randomly chosen and randomly removed from the archive, where no information of the objective function is used. Compared to DE, JADE elevates IUR after all because the indexes of the best 100p\% individuals are used. Note that the output distribution is determined only when all indexes of the best 100p\% individuals are given.

**Proposition 8.**

\[ \frac{s \sum_{i=1}^{g-1} \pi(i)}{sg \log n} \leq \text{IUR}_{JADE}(g) \leq \frac{(g - 1) \log (\frac{s}{ps}) + s \sum_{i=1}^{g-1} \pi(i)}{sg \log n}. \]  

For example, if \( n = 1024, s = 10, g = 10, p = 0.2 \), then \( 0.0621 \leq \text{IUR}_{JADE} \leq 0.1115 \).

### 4 Discussion

So far, we have discussed several algorithms and their IURs. They are summarized in Table 2.

#### 4.1 What is IUR for?

With the developments of RS, ES, PSO and DE, the IURs improve. This is not a coincidence. Elevating the information utilization ratio is the potential cause of many algorithmic improvement works in the field of heuristics. Typically algorithms in a same family share similar ways of information utilization, so elevating IUR is prone to accelerate the process of optimizing the objective function.

Figure 1 shows an example. By utilizing more information, CMA-ES is able to learn the properties of the objective function more efficiently. Similar comparisons can be made between RS and LJ, PSO and SPSO and DE and JADE unless the objective function is not in the set which these algorithms are designed to solve.

So, if you are improving a certain algorithm, and you want to check or prove that your improvement is effective and reasonable, it’s recommended that you calculate and compare the IURs of the current algorithm and your adaptation. If you are constructing a hybrid algorithm of two heuristics, it’s recommended that you calculate and check if the IUR of your algorithm is higher than each of them. If you are inventing a new heuristic inspired by some other algorithms, it’s also recommended that you compare their IURs.

It is also sometimes interesting to use IUR to guide the choice of parameters. For example, use \( \mu = \frac{4}{\lambda} \) for a fixed \( \lambda \) may be good for \((\mu, \lambda)\)-ES because it leads to the largest IUR.

However, we want to warn the readers that it’s very dangerous to compare IURs between different families of heuristics to judge or predict which one is better, because they may utilize heuristic information in extremely different ways, which means they make different assumptions of the distribution over objective functions. No algorithm is better than another under the no free lunch theorem, and only with a certain distribution over the objective functions can we discuss which algorithm is better.

Please also note that these numerical examples of the IURs are only to give the readers intuitive impressions of these equations. They should not be used to compare IURs between algorithms because the values of IUR changes with different

---

**Figure 1:** Convergence curves of \((\mu, \lambda)\)-ES and CMA-ES on two dimensional sphere function with \( \mu = 5 \) and \( \lambda = 10 \).
parameters. IURs of algorithms should be compared with regard to their expressions or using practical parameters.

### 4.2 Upper bound for comparison-based algorithms

Above examples have covered several approaches of information utilization in heuristic optimization algorithms. But the IURs of these algorithms are all very low. In fact, Theorem 2 (Upper bound for comparison-based algorithms),

If the maximal evaluation number is $m$, the evaluation values are only used to compare with each other in algorithm $A$,

$$\text{IUR}_A \leq \log_m m.$$  \hspace{1cm} (30)

**Proof.** Suppose in a certain run, the actual evaluation number is $m' \leq m$. In this case, $M$ is drawn from a set with cardinal number at most $m'$ (with $m'$ individuals all sorted), then the maximal information quantity is $H(M) \leq \log m'$ for a comparison-based algorithm. Thus $\text{IUR}_A \leq \frac{\log m'}{m' \log n}$. Note that the right hand side is a monotonically increasing function of $m'$, and $\frac{\log m}{m' \log n} \leq \frac{\log m}{m \log n}$. \hfill $\square$

This upper bound is quite low because typically $n$ is very large. For example, if there are 100 cities in a travelling salesman problem [Lawler, 1985], $n$ can be up to 100!. Not to mention continuous optimization problems.

There exist algorithms which use exact evaluation values in the searching process, such as genetic algorithm [Holland, 1975], ant colony optimization [Dorigo et al., 1996], estimation of distribution algorithms [Larrañaga, 2002], invasive weed optimization [Mehrabian and Lucas, 2006], artificial bee colony [Karaboga and Basturk, 2007], fireworks algorithm [Tan and Zhu, 2010], etc. They can achieve higher IURs, even close to 1, because the cardinal number of the set from which $M$ is drawn can be up to $n^m$. Again, it doesn’t imply these are better algorithms because they are designed to solve different problems. For example, it’s quite dangerous to use evaluation values on objective functions which fluctuate fiercely.

### 4.3 How to elevate IUR?

Roughly speaking, there are two main aspects about information utilization.

**Sampling**

In each iteration, the algorithm need to decide which solution should be evaluated. There are two main approaches: to evaluate the most promising ones (exploitation) or to evaluate the most informative ones (exploration). Usually the algorithm makes a tradeoff between them. As in ES, it’s very common to use the information of the current generation to adapt the parameters of the mutation operator in evolutionary algorithms. As in PSO, it’s very common to use the current and historical good individuals to direct search in population based algorithms. Beside the ways mentioned in the examples, there are many common ways to direct the generating of new individuals such as adaptive parameter control [Eiben et al., 1999], fitness approximation [Jin, 2005], machine learning [Battiti et al., 2008], hyper-heuristic [Burke et al., 2010], etc. They can be applied to nearly all the heuristic algorithms.

**Selection**

Selection is common in evolutionary algorithms. Candidates can be selected based on their fitness, location, diversity or luck. It is a good idea to utilize the information of the population wisely to select the next generation. There is also a tradeoff between exploration (equalism) and exploitation (elitism). In addition, external archives are frequently used in heuristics to preserve some information of eliminated individuals for further utilization.

**Remarks**

Some researchers may find the idea of utilizing more information misleading because in some algorithms, utilizing more information makes the algorithm greedier and may cause the problem of premature convergence. It’s not always the case. Undoubtedly, the most common usage of heuristic information is to lead the population toward promising locations, i.e., local minimum. But heuristic information is also vital in exploration. If no heuristic information is utilized, then sampling a point in the search space is just a random guess. Since the essence of a heuristic algorithm is to learn the objective function, both exploration and exploitation requires information utilization. The only reason why information utilization mechanisms for exploration are fewer invented is the object of exploration is vaguer. As in the case of SPSO, using local topology tends to slow down the convergence and elevate the probability to find the global optimum.

However, utilizing more information is not always better. Each kind of heuristic makes an assumption of the objective function, which is determined by the way of information utilization. If the objective function doesn’t accord with this assumption, then utilization information can be misleading.

### 5 Conclusion

In this paper, a metric for measuring the extents of information utilization in heuristic optimization algorithms is proposed, which is called information utilization ratio (IUR). Several examples are given to demonstrate the procedure of calculating IUR. These examples reveal the fact that elevating IUR is the potential cause of many algorithmic improvement works.

IUR determines the upper bound of an algorithm’s performance on a certain kind of objective function. It can be quite informative to compare IURs between algorithms which share similar ways of information utilization. Typically the algorithm with higher IUR is prone to be more efficient. IUR provides a connection between the design and the performance of heuristics. At least it’s a new perspective.

In the field of heuristic optimization, many researchers have realized the importance of information utilization implicitly or explicitly in designing or improving algorithms, but few (to the best of our knowledge) have treated this issue itself seriously. In this work, we try to make this issue more clear and ready for further studies, which is not easy. With the IUR metric, information utilization is no longer abstract. We hope IUR can guide and help researchers to improve and design heuristic algorithms more efficiently and reasonably. We also expect this concept can be extended to other fields of artificial intelligence.
References

[Auger and Hansen, 2005] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In Evolutionary Computation, 2005. The 2005 IEEE Congress on, volume 2, pages 1769–1776. IEEE, 2005.

[Auger and Teytaud, 2010] Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. Algorithmica, 57(1):121–146, 2010.

[Bäck et al., 1991] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In Proceedings of the Fourth International Conference on Genetic Algorithms, 1991.

[Battiti et al., 2008] Roberto Battiti, Mauro Brunato, and Franco Mascia. Reactive search and intelligent optimization, volume 45. Springer Science & Business Media, 2008.

[Bratton and Kennedy, 2007] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, pages 120–127. IEEE, 2007.

[Burke et al., 2010] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In Handbook of metaheuristics, pages 449–468. Springer, 2010.

[Börjesson and Ostermeier, 1996] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, pages 312–317. IEEE, 1996.

[Hansen, 2005] Nikolaus Hansen. The CMA evolution strategy: A tutorial. Vi le, 29, 2005.

[Hansen, 2006] Nikolaus Hansen. Compilation of results on the 2005 CEC benchmark function set. 2006.

[Holland, 1975] John H Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. 1975.

[Jin, 2005] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. Soft computing, 9(1):3–12, 2005.

[Karaboga and Basturk, 2007] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization, 39(3):459–471, 2007.

[Larranaga, 2002] Pedro Larranaga. A review on estimation of distribution algorithms. In Estimation of distribution algorithms, pages 57–100. Springer, 2002.

[Lawler, 1985] Eugene L Lawler. The traveling salesman problem: a guided tour of combinatorial optimization. WILEY-INTERSCIENCE SERIES IN DISCRETE MATHEMATICS, 1985.

[Loshchilov et al., 2013] Ilya Loshchilov, Thomas Stuetzle, and Tianjun Liao. Ranking results of CEC13 special session & competition on real-parameter single objective optimization. 2013.

[Loshchilov, 2013] Ilya Loshchilov. CMA-ES with restarts for solving CEC 2013 benchmark problems. In Evolutionary Computation (CEC), 2013 IEEE Congress on, pages 369–376. Ieee, 2013.

[Luus and Jaakola, 1973] Rein Luus and THI Jaakola. Optimization by direct search and systematic reduction of the size of search region. AIChE Journal, 19(4):760–766, 1973.

[Mehrabian and Lucas, 2006] Ali Reza Mehrabian and Caro Lucas. A novel numerical optimization algorithm inspired from weed colonization. Ecological informatics, 1(4):355–366, 2006.

[Shannon, 1948] C.E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27(4):623–656, Oct 1948.

[Storn and Price, 1997] Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4):341–359, 1997.

[Tan and Zhu, 2010] Ying Tan and Yuanchun Zhu. Fireworks algorithm for optimization. In Advances in Swarm Intelligence, pages 355–364. Springer, 2010.

[Wolpert and Macready, 1997] David H Wolpert and William G Macready. No free lunch theorems for optimization. Evolutionary Computation, IEEE Transactions on, 1(1):67–82, 1997.

[Zhang and Sanderson, 2009] Jingqiao Zhang and Arthur C Sanderson. JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5):945–958, 2009.