SUPER-HEURISTICS AND THEIR APPLICATIONS TO COMBINATORIAL PROBLEMS

Tak Wing Edward Lau and Yu-Chi Ho

ABSTRACT

Combinatorial problems are known to be difficult because of the shear size of the solution space and the lack of polynomial time algorithm to “solve” them. Heuristics are often devised to produce acceptable solutions in an affordable time. In this paper, we propose a method called super-heuristic that expands the capabilities of heuristics using randomization and sampling techniques. We submit that heuristics are in general strategies that map from available information of a problem instance to decisions in solution constructions/improvement. We show that it is important to utilize the information effectively in the randomization process. More importantly, the possibility of randomization around a heuristic spells out the demarcation between the roles of human and machines in complex optimization problems.

KeyWords: Combinatorial problems, heuristics, randomization and sampling techniques, man-machine interactions.

I. INTRODUCTION

Combinatorial problems are known to be difficult because of the shear size of solution space and the lack of polynomial time algorithm to “solve” them. Theoretical and application problems which demand combinatorial formulation abound. For example, in the area of control, combinatorial problems straddle between non-linear problems (Blondel and Tsitsiklis 1999) and discrete event systems (Ho 1992 and Ramadge 1989). To practically “solve” these problems, heuristics are usually devised in order to obtain computationally tractable results (see, for example, Reeves 1993). There are reasons and advantages in using heuristics:

1. One avoids doing (implicit) exhaustive enumeration of all possible (and feasible) solutions.
2. Any intuition, analysis and understanding of the underlying problem may be incorporated in the construction of heuristic solutions.

However, the disadvantages are that a heuristic does not always guarantee optimality in most situations, and the solutions obtained by a heuristic may not be easily quantified. Performance bounds are usually needed in order to gauge the quality of heuristic solutions.

Meanwhile, advances in computing technology — processing speed, memory and architecture, have offered new possibilities in the use of heuristics in combinatorial problems. A solution process involving heuristics can afford to examine a lot more solutions in much shorter time using smarter methods. In this paper, we introduce a new methodology called super-heuristic which combines both the advantages of heuristics and the capabilities of modern computing technologies. We see that this is an important area of study because it enforces an important perspective of human-machine collaboration in solving difficult design problems. Factors motivating the idea of super-heuristics are:

• Many heuristics are strong in constructing good (initial) solutions.
• In many combinatorial problems, performance evaluation of each individual solution is relatively inexpensive computation-wise (such as calculating sums and products). Therefore, it may be affordable to examine a candidate population to construct a current solution.
• There exists a collaborative relationship between computer and human in complex design problems (Ginsberg 1996). Humans are good at arriving at intuitions and
understanding, observing patterns, and gaining insights. On the other hand, the capabilities of modern computers in terms of speed and memory can easily surpass human limitations.

- The new challenge in solving complex problems is to identify and implement the tasks that are best performed respectively by human and machine (Ho 1999).

Super-heuristic is a technique which integrates the power of computing machines and heuristics derived by human. It begins with a base heuristic $H$ which constructs the base solution $x_0$. Then, we perform randomization “around” the base heuristic $n$ times to produce $n$ more solutions $x_1, \ldots, x_n$ (we shall explain what is meant by “around” later in the paper). We select the best among the $(n+1)$ solutions as the final solution $\hat{x}$. Notice that it is possible to have the final solution $\hat{x}$ again being $x_0$. However, one does not need $n$ to be very large (hence it is computationally feasible) in order to improve from $x_0$. In other words, randomization around the base heuristic can be very effective. This can be seen as follows. Suppose by randomization the probability of getting a solution $x$ which is better than that of the base solution $x_0$ is $q$, i.e.,

$$ q = \text{Prob}\{ J(x) \leq J(x_0) \} $$

where $J(\cdot)$ is the performance measure and we assume minimization, then the chance that we can improve from the base heuristic using $n$ independent randomizations is

$$ \text{Prob}\{ J(x_i) \leq J(x_0) \text{ for some } i \} = 1 - (1 - q)^n = nq $$

We can see that if $q$ was merely 0.01, we would only need $n = 100$ trials to guarantee success, which is very modest in today’s computing power. For every problem instance, the probability $q$ can be characterized by the distribution of performances and the relative position of the base solution but it also depends on how sampling is done around the base heuristic.

Other approaches using randomization on heuristics have also been proposed, including Greedy Randomized Adaptive Search Procedure (GRASP) (Feo, Resende and Smith 1995) and Heuristic-Biased Stochastic Sampling (HBSS) (Bresina 1996). In GRASP, randomization is performed based on the greedy heuristic along the way a partial solution is constructed. HBSS also considers randomizing between a greedy heuristic and purely random solutions using several classes of probability functions. These functions are based on the rankings of the entries that form the partial solution. Therefore, the probabilities are fixed regardless of what the instance is. We shall constrain our approach to these two here. First, our method accepts any good heuristic as a base heuristic.\footnote{The examples considered in this paper, however, also begin with greedy heuristics as the base.} Second, rather than using fixed probabilities for randomization, we construct probability functions based on the useful information that is available in the problem instance. In fact, we submit that in general a heuristic is a mapping from the available information to what should be constructed as partial solutions. The information may also change during the entire process of solution construction. As a result, we have a space of decision strategies to deal with. The task of randomizing around a heuristic becomes the identification of the most useful mapping in the strategy space with the base heuristic as a guiding initial point. Such a viewpoint indeed shows promising results as discussed in the examples in Sections 3 and 4.

We have mentioned earlier that performance evaluations in combinatorial problems are usually quite inexpensive and therefore it is defrayable to evaluate a population of solutions. However, many complex optimization problems also involve noisy and imprecise performance estimates and accurate measurement of them would add a hefty overhead for the heuristic implementation. To this end, one can subscribe to two tenets of a technique called ordinal optimization (Ho, Sreenivas, Vakili 1992, and Ho 1994). These two tenets suggest that it is a lot more efficient (i) to distinguish solutions based on ordinal rather than cardinal comparisons (Dai 1996), and (ii) to select a subset containing some good enough solutions rather than insisting on getting a single solution which is the best (Lee, Lau, Ho 1999). The significance of these two tenets are especially pronounced during the initial search phase of complex optimization problems, and have been found successful in many applications (e.g., Yang, Lee, Ho 1997, and Dai, Wieselthier 1999). We shall look into how ordinal optimization can be combined with super-heuristic in this paper.

Finally, it is important to emphasize that super-heuristic is a complementary framework because it is meant to work with other existing techniques to enhance the solution process:

- any analysis of a problem instance prior to the application of heuristic should be included;
- any good heuristic may serve as the base heuristic;
- in generating the candidate solutions, one may check against some pre-determined performance bounds to reject already inferior solutions;
- the output of super-heuristic can be further improved using other local search methods;
- in noisy performance evaluations, ordinal optimization extends the possibilities offered by super-heuristics.

Moreover, super-heuristic turns a combinatorial optimization problem into a strategy mapping problem involving information and decisions. This opens many research possibilities in the decision and control areas.

We shall illustrate the important ideas in super heuristic using a well known combinatorial example — the 0-1 knapsack problem. We aim at comparing the technique
with existing heuristics, and also highlighting the role of information in strategy construction. The paper is organized as follows. In Section 2, we explain the mathematical formulation for heuristic randomization via the concepts of probabilistic assignment rule and rule basis. In Section 3, we discuss the application of super-heuristic for the 0-1 knapsack problem. The results also point us to examine the properties of probabilistic assignment rule and rule basis. We further look into applying super-heuristics to the travelling salesman problem in Section 4. Comparison with another randomization technique (HBSS) will also be shown. Section 5 investigates how super-heuristic can be adapted to problems with noisy performance measure using ordinal optimization. We shall conclude in Section 6.

II. MODEL AND METHOD

There are in general two phases in solving a combinatorial optimization problem — the solution construction phase and the solution improvement phase. In the construction phase, one builds iteratively from a partial, possibly empty, solution to an admissible and complete solution. This is also referred to as tree descending method in which the top or root of the tree is an empty solution and the leaves at the bottom are complete solutions. Of course, the combinatorial nature makes the size of the tree grow very rapidly. Usually, a base heuristic corresponds to descending along the “left-most” branch of the tree. Once an initial solution is constructed, solution improvement is performed by exploiting the structure of the solution space such as exchanges of some parts of a current solution (for example, the 2-opt and 3-opt operations for the travelling salesman problem). In this paper, we are going to illustrate the idea of super-heuristic in the solution construction phase. While it is unlikely that an initial solution happens to be constructed as the optimum, a “good” initial solution is often critical in the subsequent solution improvement phase.

Let $\mathcal{X}$ be a set of $N$ objects, labeled 1, 2, ..., $N$, which are to be selected or assigned to form a (feasible) solution. The sequence by which the objects are assigned determines the final solution and hence its quality. We assume that there are $(N - 1)$ steps in the sequence and therefore only one is chosen from all the unassigned objects in each step. Let the object being assigned in the $t$th step be $y_t$. Then, the set of unassigned objects in the $t$th step is $\mathcal{X}^t = \mathcal{X} \setminus \{y_1, y_2, ..., y_{t-1}\}$.

2.1 The idea of assignment rules

A deterministic assignment rule (DAR) is comprised of $(N - 1)$ mappings that specify the sequence of assignments. Precisely, a DAR is $r = \{r^1, ..., r^{N-1}\}$, and the mapping $r^t$ assigns an object at the $t$th step based on the current information available for all unassigned objects:

$r^t : I^t \to \mathcal{X}^t, t = 1, ..., N - 1$

where $I^t$ is the information set at the $t$th step. The concept of information set is central to the design of heuristics. Some well known examples are

- Nearest Neighbor (NN) rule for the travelling salesman problem (TSP) in which the information is contained from the distance matrix of the instance. The output of each mapping is the city (object) associated with the smallest value among the entries in row of the last visited city (assigned object).
- Shortest Due Date (SDD) rule for the job shop scheduling problem in which the information is the list of all due dates in ascending order and the first in the list is assigned.

Usually, some physical attributes of the unassigned objects will go into the information set but it requires some human ingenuity to determine what and how these attributes are processed as well as how the information should be combined in the assignment mappings. We assume in this paper that the information is summarized into a scalar quantity $\alpha_j$ for each object $j$, i.e., $I^t \subseteq \mathcal{R}^{N-1}$.

In many combinatorial problems, a simple monotone ordering of the information set $I^t = \{\alpha_{j_1}, j_1 = 1, ..., N - t + 1\}$ is commonly used (the bracketed subscript indicates that the entries are ordered). The assignment is made on the object based on either the highest or lowest rank in the ordering. These rules are often termed “greedy rules” or “myopic policies.” However, it may not be optimal to consider monotone ordering for all instances of a problem. There are two possibilities. One is to design a non-monotonic mapping using entries in the information set for each problem instance. The other possibility is to deviate from the monotone ordering using chance variables. The latter brings us to the key component of super-heuristic — probabilistic assignment rule.

A probabilistic assignment rule (PAR) is comprised of $(N - 1)$ mappings that specify a sequence of probability laws for objects to be assigned. A PAR is $\rho = \{\rho^1, ..., \rho^{N-1}\}$, and the mapping $\rho^t$ takes the information available at the $t$th step and outputs a probability density by which an object is drawn:

$\rho^t : I^t \to \mathcal{P}(\mathcal{X}^t), t = 1, ..., N - 1$

or for $f \in I^t$, $\rho^t(f) = \rho^t$ where the probability density $\rho^t \in (\mathcal{P}(\mathcal{X}^t)$ is defined on $\mathcal{X}^t$. Therefore, the output of each $\rho^t$
is a probability vector on the \((N - t + 1)\) dimensional simplex, i.e.,
\[
P' = \rho'(F) = (p'_1, p'_2, \ldots, p'_{N-t+1})
\]
such that \(\sum_j p'_j = 1\) and \(p'_j \geq 0\), \(\forall j\).

Notice that the objects associated with the coordinate axes do change for each \(r\) to account for objects already selected in steps preceding the current one. In addition, we have not specified the arrangement between the coordinate axes and the information set. Without loss of generality, we adopt the following convention in this paper:

**Assumption AXES:** The ordering of the coordinate axes in \(P'(\mathcal{X}')\) follows the monotone ordering as the entries in the information set \(\{\alpha'_{t1}, \alpha'_{t2}, \ldots, \alpha'_{t(N-t+1)}\}\).

Therefore, the object associated with \(\alpha'_{tj}\) will have a probability of \(p'_j\) being picked in the \(k\)th step. The monotone ordering here can be either ascending or descending.

**Examples of PAR:** Consider the \(r\)th step where the objects \(\{y_1', \ldots, y_{t-1}'\}\) have already been assigned. The following are possible candidates in \(P'(\mathcal{X}')\):
- \((1.0, 0.0, \ldots, 0.0)\) — this is often referred to as the “greedy” rule
- \((0.0, 1.0, \ldots, 0.0)\) — this deterministically chooses the object in the second rank
- \((0.5, 0.5, \ldots, 0.0)\) — this picks with equal chance the objects in the first two ranks
- \((\frac{1}{N-t+1}, \ldots, \frac{1}{N-t+1})\) — this picks with equal chance all unassigned objects

In fact, one may consider that the generation of PARs creates by itself a space of heuristics characterized by the probability assignments.

2.2 The algorithm

We can now formally state the super-heuristic method as an algorithm. It begins by writing a base heuristic \(H\) as a deterministic assignment rule \(r\) which generates a base solution \(x_0\). Then, a probabilistic assignment rule \(\rho\) based on \(r\) is also determined. Together with a random device, \(\rho\) will produce \(n\) more solutions \(x_1, \ldots, x_n\). All the \((n + 1)\) solutions are compared and the best among them, call it \(\hat{x}\), is selected as the output of the algorithm, and can be further improved by some local search methods. Schematically, the algorithm is summarized in Fig. 1. Notice from Fig. 1 that the superheuristic algorithm naturally admits parallel implementation by either SIMD or MIMD architecture. Each solution generated by \(r\) and \(\rho\) can be designated to a single processor while the selection is done via communications between the processors.

How do we determine \(\rho\) from \(r\)? This brings us to a more general design of the super-heuristic algorithm. We can generate a number of \(\rho\)'s, say \(\rho_1, \ldots, \rho_n\), and each of them is applied as aforementioned. The modified schematic is shown in Fig. 2. The \(R\) different \(\rho\)'s can be sampled uniformly using a set of basis vectors called the rule basis, as discussed below.

2.3 The idea of rule basis

Since \(P'(\mathcal{X}')\) is an \((N - t + 1)\)-dimensional simplex, therefore, the output of \(\rho\) is completely specified using a choice of rule basis — a set of linearly independent vectors on the simplex. Naturally, the canonical basis with \(e_j = [0, 0, \ldots, 1 \text{ (jth position)}, \ldots, 0]' \in \{0, 1\}^{N-t+1}\) can generate all points on the \((N - t + 1)\) dimensional simplex. That is, any \(P \in P'(\mathcal{X}')\) can be written as \(P = \beta_1 e_1 + \beta_2 e_2 + \ldots + \beta_n e_n\), where \(\beta_j \in \{0, 1\}, j = 1, \ldots, N - t + 1\). However, the canonical basis may not be the most useful rule basis. The following is another example of rule basis \(B' = [b_1, b_2, b_3, b_4, b_5]'\) for 5 objects:
monotone basis, denoted by $\tilde{B}^{N-t+1}$, if the entries in the basis vectors are $1/k$ for the first $k$ elements in the $k$th vector, and zero otherwise:

$$\tilde{b}_{kj} = \begin{cases} 1/k & \text{for } j \leq k, \\ 0 & \text{otherwise} \end{cases}$$

for $k = 1, \ldots, N-t+1$. It can be easily seen that the above basis $B^5$ is a monotone basis $\tilde{B}^5$ in the 5th dimension. The reason why this is termed “monotone” is that for $\beta_j \in [0,1]$, the linear combination $\beta_1 b_1 + \beta_2 b_2 + \ldots + \beta_{N-t+1} b_{N-t+1}$ only produces vectors with components $p_1 \geq p_2 \geq \ldots \geq p_{N-t+1}$ on the simplex. Therefore, not all points on the simplex are reached by the monotone basis. When there are only three objects, the region inscribed by the monotone basis $\tilde{B}^3$ is depicted in Fig. 3.

Notice that under Assumption AXES, a monotone basis indeed favors the monotone ordering of the information set (e.g., the greedy heuristics). For many heuristics, entries in the information set are usually derived from some useful attributes which indicate the “preferability” of unassigned objects. A greedy heuristic, for example, follows strictly this preferability ordering. While the optimal solution does not always obey exactly a monotone ordering with respect to the information set, it is often advantageous to select from the unassigned objects using probabilities close to the monotone ordering. Similarly, there are objects which are unlikely be included in the solution because they are apparently undesirable, and should have little chance to be picked in the assignment rule. Therefore, monotone basis keeps the “flavour” of a good heuristic but at the same time allows for deviations so as to capture better solutions.

Finally, both canonical and monotone bases are static in the sense that the entries in the basis vectors are fixed as $1/k$ in the $k$th vector. Consider the following basis vectors in the $n$th step

$$d'_1 = 1/A'_1 [ \alpha'_{[1]}, 0, 0, \ldots, 0 ]'$$
$$d'_2 = 1/A'_2 [ \alpha'_{[2]}, 0, 0, \ldots, 0 ]'$$
$$d'_3 = 1/A'_3 [ \alpha'_{[3]}, \alpha'_{[2]}, 0, \ldots, 0 ]'$$

where $A'_i = \sum_j \alpha'_{ij}$ is the normalization factor. In this kind of basis, probability assignments depend directly on the relative “strength” of the unassigned objects in terms of the information $\alpha'_{ij}$. The superscript $\prime$ indicates that the entries vary from step to step. The rationale for this construction is that the basis vectors should reflect the relative “strength” of the information instead of weighing equally on the unassigned objects. For example, suppose that for only two objects A and B whose information is respectively $\alpha'_{ij} = 99$ and $\alpha'_{iB} = 1$ and that A is more preferred than B. Then, it may be more plausible to generate probabilities closer to (0.99, 0.01) rather than (0.50, 0.50). Due to the presence of the information terms $\alpha'_{ij}$’s, we shall refer this kind of basis to as information dependent basis or IDB. The entries in an IDB can be arranged in many fashions. For example, the basis $D = [d'_1, d'_2, \ldots, d'_n]$ defined above is a first degree IDB, and a second degree IDB will have entries $(\alpha'_{ij})^2$, properly normalized. An $\eta$-degree exponential IDB will have basis vectors

$$d'_{ij} = 1/A'_j [ e^{\eta p_{[1]}}, \ldots, e^{\eta p_{[n]}}, 0, \ldots, 0 ]'$$

where $\eta$ is a scaling factor. Various mathematical constructions (polynomial, logarithmic, etc.) using the information set are possible and will produce different results in the super-heuristic. In general, the higher the degree in an IDB, the closer the basis vectors to $(1, 0, 0, \ldots, 0)$, i.e., the base heuristic. These bases will be compared in the subsequent sections in this paper.

To summarize, a super-heuristic can be coded by a 3-tuple S-H($B^n$, $R$, $n$) where $B^n$ is the choice of basis (of degree $\eta$), $R$ is the number of PARs generated in $B$, and $n$ is the number of solutions sampled from each of the $R$ PARs. The ideas of super-heuristic and rule bases will be demonstrated more concretely using the 0-1 knapsack problem in the next section. We compare the results of PARs generated by different bases against the base heuristic. We also examine properties pertaining to the implementation of PARs. Further results on the travelling sales problem (TSP) and comparison with another technique (HBSS) will be shown in the section following next. In general, super-heuristic produces better results than those obtained by known benchmark heuristics.
III. EXPERIMENTAL RESULTS

3.1 The knapsack problem

A 0-1 knapsack problem KS(N, M, h, W, c) with N objects and M constraints is defined by:

$$\max_{x \in \{0, 1\}^N} hx$$

$$s.t. \ W x \leq c$$

where h is a (1 × N) vector of “benefits,” W is a (M × N) matrix of “weights,” c is a (M × 1) vector of “capacities.” The decision vector x is (N × 1) of 1’s and 0’s, and the solution space is of cardinality $2^N$ (before constraints). The knapsack problem is known to be NP-complete. It is also a general formulation for problems in which resources are limited and decisions have to be made in order to satisfy the capacity constraints while maximizing benefits. Some examples include resource allocation problems (Ozden 1988) and sensor fusions for signal detection in limited bandwidth (Pao 1995). The wide applicability of this formulation and its variants have attracted much attention using, for example, neural networks (Ohlsson, Peterson and Soderberg 1992) and simulated annealing (Drexl 1988). Using, for example, neural networks (Ohlsson, Peterson and Soderberg 1992) and simulated annealing (Drexl 1988). We are here to compare the idea of super-heuristic against a popular base heuristic — the greedy rule.

3.1.1 Information set and base heuristic

The information set can be derived as follows. For each object i we compute $\alpha_i = h_i/w_{ij}$ (for single constraint), and we call this number the $\alpha$-ratio of object i. The higher the $\alpha$-ratio of an object, the more desirable an object is to be included in the knapsack because of the higher benefit and less constraint upset. Therefore, we construct a monotone ordering of the $\alpha$-ratios in descending order $\alpha_{[1]} \geq \alpha_{[2]} \geq \ldots \geq \alpha_{[m]}$.

A greedy rule is to include as many objects as possible in the $\alpha_{[j]}$’s (j = 1, ..., N). In other words, it is to find the largest k such that $\alpha_{[1]} \geq \ldots \geq \alpha_{[k]}$ and the constraint Wx ≤ c is not violated. The greedy rule is a reasonable and intuitive heuristic but it cannot guarantee optimal solution (s) for general 0-1 knapsack problems (Syslo, Deo, Kowalik 1983). In this paper, we employ the greedy rule as the base heuristic.

3.1.2 Experiment settings and performance measures

Without loss of generality, we may set the $h_i$’s and w_{ij}’s to be generated uniformly between 0 and 1, i.e., $h_i \sim U[0, 1], w_{ij} \sim U[0, 1]$ for all i = 1, ..., N and j = 1, ..., M. As for the values $c_1, \ldots, c_M$, for non-triviality reasons, we must have

$$\min_{i} w_{ij} \leq c_j \leq \sum_{i} w_{ij}$$

for at least one of the constraints. Typical test cases set the c_j’s around N/4. Using these settings, we can generate many problem instances for testing assignment rules. The set of all possible instances is the instance space, which has infinitely many elements.

To gain some understanding about the super-heuristic method, we compare probabilistic assignment rules (PARs) against the base heuristic, i.e., the greedy rule. Notice that the greedy rule can be considered as a degenerate PAR. We uniformly sample R different $\rho$’s using canonical, monotone and information driven basis. All the $\rho$’s as well as r are tested on S instances generated from the instance space. In Case #1, #2 and #3 below, the performance of each PAR is calculated by averaging its outputs of the $S=1,000$ problem instances. This is also the performance measure considered by Chen and Dai in their study of heuristics (Chen, Dai 1999). For the first two cases, we aim at contrasting results using monotone basis and the base heuristic. We also compare the effect of using different number of realizations (the value of n). In Case #4, we consider the percentage of instances for which a PAR outperforms the base heuristic.

3.1.3 Results

Case #1: N = 10, M = 1, S = 1000, n = 100, R = 500, $B = \text{Monotone}$

We begin with a smaller sized problem for which we can exhaustively locate the optimal solution. We have plotted the results of the 500 PARs and also the greedy rule solutions in Fig. 4. Each PAR is run with n = 100 realizations (randomized solutions) and the best output is used for comparison. We see that the greedy rule is dominated by all the PARs which shows that deviating from the greedy rule will in general produce

Fig. 4. Greedy rule is outperformed by all PARs.
better solutions.

Case #2: \( N = 10, M = 1, S = 1000, n = 20, R = 500, B = \text{Monotone} \)

Here we let each rule to have only \( n = 20 \) realizations instead of 100 in the previous case. The results are shown in Fig. 5. We observe that most PARs still perform better than the greedy rule. Recall that if the chance of improving (maximizing) from a greedy rule using a PAR is \( q = \text{Prob} \{ J(\text{PAR}) > J(\text{Greedy}) \} \) (cf. Section 1), then the probability of success using \( n \) independent randomizations is \( 1 - (1 - q)^n = nq \). This shows that the greedy rule can be improved quite effectively for \( n = 20 \). At a value of \( q \approx 0.05 \), it is possible to conceive that the greedy rule is 95\% effective and therefore it actually serves as a very good base heuristic.

Case #3: \( N = 10, M = 1, S = 1000, n = 100, R = 500, B = \text{Canonical vs. Monotone} \)

In this case, we show that the monotone basis outperforms the canonical basis (Fig. 5). The canonical basis does not follow the monotone ordering of the \( \alpha \)-ratios specified by the base heuristic. Therefore, we see that more than half of the PARs generated by the canonical basis perform worse than the greedy rule.

Case #4: \( N = 10, M = 1, S = 1000, n = 20, R = 500, B = \text{Canonical, Monotone & IDB} \)

We compare three different rule bases: canonical, monotone, and a first degree IDB. Instead of measuring the average performances, we choose the percentage of instances that a PAR generated by each basis performs better than the greedy rule (Fig. 7). We see that the information driven basis works better than both canonical and monotone bases. This is due to the fact that an IDB makes use of the \( \alpha \)-ratios in the information set.

### 3.2 Properties of basis in super-heuristic

We have seen in the previous section how different bases compare with each other when measured against the base heuristics. In this section, we analyze these bases more closely.

Since there are infinitely many PARs that can be generated by a certain basis, we choose to identify a set of characteristic positions of a basis for analysis purpose. The characteristic positions of a basis are a set of vectors defined by

\[ \mu_k^t = \frac{1}{k} \sum_{j=1}^{k} b_j^t \quad \text{for} \quad k = 1, \ldots, N - t + 1. \]

In other words, \( \mu_k^t \) is the centroid of the first \( k \) basis vectors. We can visualize schematically these vectors by an \((N - t + 1)\) polygon as shown in Fig. 8. For instance, the centroids in the 5th dimensional monotone basis are

\[
\begin{align*}
\mu_1^5 &= [1, 0, 0, 0, 0]’, \\
\mu_2^5 &= [3/4, 1/4, 0, 0, 0]’, \\
\mu_3^5 &= [11/18, 5/18, 1/9, 0, 0]’, \\
\mu_4^5 &= [25/48, 13/48, 7/48, 1/16, 0]’, \\
\mu_5^5 &= [257/300, 77/300, 47/300, 9/100, 1/25]’. 
\end{align*}
\]
Notice that these characteristic positions are only some particular PARs that deviate from the base heuristic but they are indicative of the performances of a basis. We have measured the performance of the characteristic positions of three bases: canonical, monotone and first degree IDB for the 0-1 knapsack problem with 10, 40 and 75 objects. (There are only 10 characteristic positions for $N = 10$ problems.) The comparisons of these bases are shown in Fig. 9, 10, and 11. Again, we are measuring the percentage of instances that PARs from a basis performing better than the base heuristic. As seen in Fig. 9, the lower characteristic positions of the canonical basis are stronger than the rest of the positions. Meanwhile, in Fig. 10 and 11, there are many more strong characteristic positions. This is because the subspace spanned by the monotone and IDB bases are more focused. In fact, there is a region in the probability space which gives the best results for randomizing around a base heuristic. This region is "near" the base heuristic. It is in general difficult to determine precisely this region for general problems. The design of effective rule basis is to match this region as close as possible.

IV. FURTHER RESULTS

4.1 The travelling sales problem

The travelling salesman problem TSP($N, W$) is the most widely studied combinatorial optimization problem (Johnson, McGeoch 1997). It is characterized by $N$ cities and a distance matrix $W$ of all bi-directional distances between the cities. The objective is to find the shortest tour that traverses all $N$ cities only once and return to the starting city (i.e., a Hamiltonian circuit). In this section, we consider only the symmetric problems such that the entries in $W$ are $w_{ij} = w_{ji}$ and the diagonal elements are zero.

4.1.1 Information set and base heuristic

The information set of a TSP is just the distance matrix $W$. There are a number of base heuristics for constructing tours ([reference]). We choose the nearest neighbor (NN) rule as the base heuristic. Suppose at the $r$th step, the last visited city was $y_{r-1} = i$. Then, the NN rule specifies that the $r$th city to be visited is

$$\arg \min_{j \in A_i} w_{ij},$$

where $A_i = \{y_1, \ldots, y_{r-1}\}$.

In order to conform with the formulation in this
paper, we define the information set to be \( I_t = \{ \alpha_1^{[j]}, \alpha_2^{[j]}, \ldots, \alpha_{N-1}^{[j]} \} \) where \( \alpha_1^{[j]} \) is negative of the distance to the \( j \)th nearest neighbor from \( y_t \). This allows for construction of exponential IDB’s that give higher probabilities to unvisited cities of smaller distances from a current city, i.e.,

\[
\begin{align*}
  d_t^i &= 1 / A_t^{[1]} \left( e^{\eta \alpha_1^{[1]}}, \ldots, e^{\eta \alpha_k^{[1]}}, 0, \ldots, 0 \right) \\
  &= 1 / A_t^{[1]} \left( e^{-\eta w_i^{[1]}}, \ldots, e^{-\eta w_i^{[k]}}, 0, \ldots, 0 \right)
\end{align*}
\]

where \( w_i^{[j]} \) is the \( j \)th nearest admissible distance from city \( i \), \( \eta \) is a scaling factor, and \( A_t^{[1]} = \sum_j e^{\eta \alpha_j^{[1]}} \).

4.1.2 Results using randomly generated instances

We have generated \( S = 1,000 \) random instances of 10-city and 30-city TSP from the unit square. The number of samples \( n = 20 \) is the same for all cases. We measure the percentage of instances that a PAR outperforms the base heuristic for 5 different bases: canonical, monotone, and exponential IDB of degrees \( \eta = 1, 7, \) and 20 (cf. Section 2.3). We have also recorded the percentage of value improvement from the base solution. They are shown respectively in Fig. 12 and 13 for the 10-city problem and Fig. 14 and 15 for the 30-city problem. (The legends in Fig. 12-15 are: “Cn” for Canonical, “Mn” for Monotone, and “E1” for \( \eta = 1 \), etc.)

4.1.3 Results using the TSPLIB cases

We have compared the NN heuristic against super-heuristic for a number of cases drawn from the TSPLIB. The TSPLIB is a resourceful library for the travelling salesman problem and contains an extensive collection of test cases for symmetric, non-symmetric and other related TSP problems.\(^5\) Table 1 shows that the NN heuristic is

<table>
<thead>
<tr>
<th>Case</th>
<th>NN</th>
<th>SH</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berlin52</td>
<td>8181</td>
<td>7843</td>
<td>7542</td>
</tr>
<tr>
<td>ST70</td>
<td>796</td>
<td>739</td>
<td>675</td>
</tr>
<tr>
<td>PR76</td>
<td>130921</td>
<td>123591</td>
<td>108159</td>
</tr>
</tbody>
</table>

Table 1. Results from TSPLIB instances.

\(^5\)The TSPLIB can be accessed online at [http://softlib.rice.edu/softlib/tsplib](http://softlib.rice.edu/softlib/tsplib).
always outperformed by the super-heuristic but the results are still to be improved by local search method. The results in Table 1 are calculated using $R = 5$ PARs, each with $n = 10$ realizations.

4.2 Comparison with HBSS

Finally, we compare our results against the Heuristic Based Stochastic Sampling (HBSS) technique (Bresina 1996). HBSS uses several classes of probability functions including logarithmic, polynomials of various orders, and exponential. The assignment of probabilities for the unassigned objects is based on the respective ranks. For example, the most effective probability function reported in (Bresina 1996) is exponential which assigns these values to the monotone ordering of unassigned objects

$$p_k = e^{-k} / \sum_{j=1}^{N-t+1} e^{-j}$$

where $k = 1, \ldots, N - t + 1$. Table 2 summarizes the comparison results for 0-1 knapsack problem of different sizes. The comparison is based on 100 realizations for HBSS and five PARs each with 20 realizations. The number of problem instances is 1,000 and the exponential class of IDB is used as well.

From Table 2, we see that randomizing around the greedy heuristic produces better results using HBSS but the super-heuristic idea further improves the performances. This is due to the fact that an IDB is able to utilize the information in a problem instance (instead of using only the ranks) and explore other randomizing probabilities within the basis. However, since super-heuristic always takes the best solution generated from a basis, it is generally worth the exploration. In addition, we see in this comparison that it is important to identify the most effective form of basis to obtain better results.

V. MODIFICATIONS TO NOISY MEASUREMENTS

The objective value of a given solution in either the 0-1 knapsack problem or the TSP can be precisely determined, usually as a sum of the multiples of the parameters in the problem instance. In this section, we investigate another scenario when the evaluation of objective value is subjected to additive noise. For example, in the 0-1 knapsack problem, we have

$$\max_{x \in \{0, 1\}^N} J(x) = hx$$

$$s.t. Wx \leq c$$

However, we cannot accurately calculate the true objective value $J(x) = hx$ but an observed value $\tilde{J}(x) = hx + \xi$ where $\xi$ is a gaussian random variable $N(0, \rho^2)$. One possibility is to obtain an estimate $\tilde{J}(x)$ by averaging, say $L$, independent evaluations $z_1, z_2, \ldots, z_L$, i.e., $\tilde{J}(x) = \sum_{l=1}^{L} z_l / L$. However, the accuracy of $\tilde{J}(x)$ is at best improving at a rate proportional to $1/L$ (Kushner, Clark 1978). In other words, one order of accuracy improvement requires at least two orders of independent evaluations. For the super-heuristic algorithm stated in Section 2, we would have selected from the $n$ solutions generated by a PAR one which has the highest value of $\tilde{J}(x)$. Figure 16 plots how PARs at different characteristic positions outperform the base heuristic under various levels of noise-to-signal ratios (from 0.0 to 10.0). We see that the comparison between PARs and base heuristic deteriorates as noise increases.

On the other hand, we can modify the super-heuristic algorithm only slightly to accommodate the noisy measurement situation. This modification is due to two tenets of the ordinal optimization technique (Ho, Sreenivas, Vakili 1992 and Ho 1994):

1. It is exponentially efficient to identify good solutions using ordinal comparison of the observed performances (asking whether $A > B$ rather than $A - B = ?$) (Dai 1996);
2. It is also exponentially efficient to select a number of solutions to match some good enough solutions using observed performances (Lee, Lau, Ho 1999).

Therefore, in the super-heuristic implementation, we can run a few samples $z_i$ for each solution $x$ to obtain a

<table>
<thead>
<tr>
<th>$N$</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDB</td>
<td>51.18</td>
<td>60.00</td>
<td>66.82</td>
<td>70.54</td>
<td>73.28</td>
</tr>
<tr>
<td>HBSS</td>
<td>49.96</td>
<td>59.54</td>
<td>65.46</td>
<td>67.46</td>
<td>71.88</td>
</tr>
</tbody>
</table>

In Bresina (1996), for $k < 6$, the probabilities are lumped together in a single number.

---

7 We do assume the constraint $Wx \leq c$ can be accurately measured.
rough estimate $J(x)$ for comparison (Tenet 1). Instead of using the best observed solution as the output, we select a few, say $s$, solutions with the $s$ highest $J(x)$’s (Tenet 2). Subsequently, more averaging can be done on these $s$ selections in order to identify the most promising solution. In this fashion, the super-heuristic algorithm can be coded as $S\text{-}H(B, R, n, s)$ where $B, R, n$ are respectively the basis, number of PARs and number of realizations, and $s$ is the selection size. The determination of $s$ depends on the underlying noise-to-signal ratio and desired matching probabilities (Lau, Ho 1996). Figures 17 and 18 show at two different noise-to-signal ratios (1.0 and 0.5) how different values of $s$ help bring up the percentage values shown in Fig. 16.

VI. CONCLUSIONS

We have proposed in this paper a new method called super-heuristic for combinatorial optimization problems. The method exploits the advantages of using a good heuristic and the benefits of randomizing around it to produce better solutions. Improvement comes in at the expenses of random number generation and deployment of probability laws. An important point of view is to recognize a heuristic as a strategy mapping between the information available in a problem instance to the decisions of constructing a solution. Therefore, the problem is turned to one searching in the space of randomized heuristics in which the base heuristic is also a member. We have proposed a simple model to describe this randomization space.

By means of two examples, the 0-1 knapsack problem and the travelling salesman problem, we have illustrated the generation of randomized heuristics using the idea of rule basis. A number of these bases are examined and compared. We have shown how an information dependent basis (IDB) can be constructed by means of the attributes in a problem instance and that IDBs indeed produce better results. More specifically, there exists a region in the probability space that renders the best improvement from the base heuristic. We aim at capturing this region by constructing some useful bases, and this warrants further theoretical efforts.

Super-heuristic can be seen as a probabilistic way of solution construction for combinatorial problems. As mentioned in the beginning of Section 2, this can be understood as tree descending method and the base heuristic produces the solution corresponding to the “left-most” branch of the tree (i.e., purely depth-first approach). Besides randomizing a heuristic, there are also other deterministic methods such as depth-bounded discrepancy search (DDS) that systematically deviates from the base heuristic in descending the solution tree (Walsh 1997). The rationale is that it is more likely to commit mistakes in the top rather than in the bottom of the tree. In this light, the use of effective bases and generation of probability assignment rules $\rho_1, \rho_2, \ldots, \rho_{N-1}$ in super-heuristic aim at minimizing the mistakes in the each of the $(N-1)$ levels of descending the tree using the information available in the tree.

In addition, we have investigated a scenario of noisy performance evaluations for the 0-1 knapsack problem. We have shown how the super-heuristic algorithm can be combined with techniques in ordinal optimization which result in better identification of solutions within affordable computational resources.

Finally, it is important to emphasize again the motivation of super-heuristics, i.e., combining heuristics with randomization. Devising good heuristics amounts to understanding of a problem, analysis of underlying structure, and recognition of useful patterns. It is an endeavor that by virtue of human wisdom, creativity and experiences a designer would come up with better heuristics. On the other hand, randomization and certain level of brute force exploration are better suited for powerful machines because of the faster processing speed, available memories and ingenious architectures — and the ever-advancing computing technologies. There exist different tasks in complex problem solving that human and machine can best
contribute to the final goal (Ginsberg 1996). It is the synergy between human intelligence and machine capabilities that push forward the solution process (see Fig. 19). Such a demarcation between the roles assumed by human and machine evolves constantly with respect to technology, and, perhaps even more so, to theoretical advances (Ho 1999). Therefore, it urges both theoreticians and computational practitioners to work at more sophisticated levels and serve a more collaborative relationship to each other. This is why we promote super-heuristic as a complementary framework on one hand, and a means to realize such a division of tasks on the other.

ACKNOWLEDGEMENTS

The authors would like to thank Professor Feng-Tse Lin and Wheeler Ruml for their valuable discussions and references.

REFERENCES

Tak Wing Edward Lau is presently a Post-doctoral fellow in the Division of Engineering and Applied Sciences at Harvard University where he also received his Ph.D. in Applied Mathematics in 1997. He was a Chu scholar at Boston University where he earned his B.S. degrees and M.S. degrees, both in Manufacturing Engineering. His research interests include simulation, optimization and analysis of manufacturing systems, algorithm design and software implementation with emphasis on industrial applications.

Yu-Chi (Larry) Ho received his S.B. and S.M. degrees in Electrical Engineering from M.I.T. and his Ph.D. in Applied Mathematics from Harvard University. Except for three years of full time industrial work he has been on the Harvard Faculty. Since 1969 he has been Gordon McKay Professor of Engineering and Applied Mathematics. In 1988, he was appointed to the T. Jefferson Coolidge Chair in Applied Mathematics and Gordon McKay Professor of Engineering at Harvard and as visiting professor to the Cockrell Family Regent’s Chair in Engineering at the University of Texas, Austin.

He has published over 120 articles and three books, one of which (co-authored with A.E. Bryson, Jr.) has been translated into both Russian and Chinese and made the list of Citation Classics as one of the most referenced works on the subject of optimal control. He is on the editorial boards of several international journals and is the editor-in-chief of the new international Journal on Discrete Event Dynamic Systems. He is the recipient of various fellowships and awards including the Guggenheim (1970) and the IEEE Field Award for Control Engineering and Science (1989), the Chiang Technology Achievement Award (1993). He is a fellow of IEEE, a Distinguished Member of the Control Systems Society, and a member of the U.S. National Academy of Engineering. In addition to serving on various governmental and industrial panels, and professional society administrative bodies, he was the President of the IEEE Robotics & Automation Society in 1988 and co-founder of Network Dynamics, Inc., a software firm specializing in industrial automation.

His research interests range from differential games, information structure, multi-person decision analysis, to incentive control, and since 1983, exclusively in discrete event dynamic systems, perturbation analysis, and manufacturing automation.