Space Exploration and Global Optimization for Computationally Intensive Design Problems: A Rough Set Based Approach

Songqing Shan G. Gary Wang[*]
Dept. of Mechanical and Industrial Engineering
The University of Manitoba
Winnipeg, MB, Canada R3T 5V6
Tel: 204-474-9463 Fax: 204-275-7507
Email: gary_wang@umanitoba.ca

Abstract

Modern engineering design problems often involve computation-intensive analysis and simulation processes. Design optimization based on such processes is desired to be efficient, informative, and transparent. This work proposes a rough set based approach that can identify multip,le sub-regions in a design space, within which all of the design points are expected to have a performance value equal to or less than a given level. The rough set method is applied iteratively on a growing sample set. A novel termination criterion is also developed to ensure a modest number of total expensive function evaluations to identify these sub-regions and search for the global optimum. The significances of the proposed method are two folds. First, it provides an intuitive method to establish the mapping from the performance space to the design space; given a performance level, its corresponding design region(s) can be identified. Such a mapping can be used to explore and visualize the entire design space. Second, it can be naturally extended to a global optimization method. It also bears potentials for more abroad applications to problems such as robust design optimization. The proposed method was tested with a number of test problems and compared with a few well-known global optimization algorithms.

Keywords: rough set, design optimization, space exploration, global optimization

1. Introduction

In product design, design engineers not only want the final optimal solution, they would also prefer a more informative process that can help them quickly and/or interactively search for the optimum. In another word, a transparent and informing search process, rather than a blind process, is desired. In addition, due to many practical limitations, a single crisp optimum might not be a good solution. Engineers want to know in such cases what will be the other close-to-

[*] Corresponding author

optimum alternatives. A set of attractive design alternatives, rather than a single optimum, is thus preferred. All these demands can be transformed into one question: given a set of design requirements, how to find those satisfactory and / or best design alternatives? Ullman (2002) described 17 "wishes" for an ideal mechanical engineering design support system. The first two wishes center on exactly the above needs, i.e, a design support system should give the engineers the ability to work from function (design requirements) to geometry form of a product (product design).

The recent design visualization methods, such as the Computer Steering and Visual Steering methods, strive to enable engineers to interact with the search process (Eddy and Lewis 2002, Winer and Bloebaum 2002a, 2002b). One major problem that Eddy and Lewis (2002) have encountered is the mapping from the performance space to the design space. In another word, given some performance level, the problem is how to find many design alternatives that will suffice.

Traditional optimization processes start from a starting point (design) and check its performance; the process iterates until the optimum is found. This process is thus a forward search method from the design space to the performance space. Can we search backwards from the performance space to identify the corresponding design space(s)? In practice, it is natural to intuitively reduce the design space gradually to a small area (Reddy 1996). However, most of these methods are *ad hoc* and problem dependent.

In recent years in the area of Multidisciplinary Design Optimization (MDO), a branch of researches aims at developing methods that can gradually reduce the design space in an optimization framework. The design space, defined by the combination of bounds of all variables, is a hyper-box in an *n*-dimensional space. Such a hyper-box, as first given by design engineers, reflects the scope of search and significantly influences the overall optimization time. Two types of design space reduction schemes are seen in the literature. One is to reduce the dimensionality of the design space by reducing the number of design variables. However, dimensionality is difficult to reduce, especially for multidisciplinary design problems (Koch et al. 1999). The other type of design space reduction seeks to reduce the size of the design space while assuming the dimensionality cannot be further reduced. A detailed review on this research direction can be found in authors' previous work (Wang and Simpson 2002). Most, if not all, of those methods are developed to search for the design optimum, by taking the advantage of sampling, space reduction, and the interplay between the two. They are essentially forward searching methods.

In the field of global optimization, multi-start methods such as clustering (Boender et al. 1982, Jain and Agogino 1993) divide a big design space to small regions so that each region contains a local optimum. However, these methods are meant to search for local optima, with no regard to if the local optima satisfy a certain performance requirement. That is, some local optima may have unacceptable performance values. Therefore, these global optimization methods are not developed to establish the aforementioned mapping from performance to design. A recent detailed review is given by Neumaier (2001) on global optimization methods.

It is to be noted also that today's engineering design often involves computationally intensive analysis and simulation processes such as finite element analysis (FEA), computational fluid dynamics (CFD), and so on. For instance, one crash simulation of a full passenger car takes 36-160 hours to compute, according to engineers at For Motor Company (Gu 2001). Also a global design optimum is always more attractive than a local optimum if the computation cost is acceptable.

In summary, a systematic and domain-independent method that establishes the mapping from performance to design space is needed. This work addresses this need by proposing a systematic method that can identify a region or regions in a design space within which all of the design points are expected to have function values less than or equal to a given performance level. If the given performance level is given as a desired target, continuous and / or discontinuous regions in the design space can be obtained; any point within is expected to have satisfactory performance. If desired, engineers can explore further in those regions looking for the global optimum. This work also bears in mind the goal of reducing the total number of expensive function calls. Because this work is based on a new mathematical concept, rough set, and to the best of the authors' knowledge it is the first time that the rough-set is introduced into the mechanical design area, the related rough set theory is first introduced and described in the following section.

2. Related Rough Set Theory

Rough set theory was developed by Pawlak (1982) in the early 1980's. Its main goal is to synthesize approximation of concepts from the acquired data. It deals with the classificatory analysis of data. Rough sets have been successfully applied in medicine (Pawlak et al. 1986) finance (Golan and Ziarko 1995), telecommunication (Czyzewski 1997), material analysis (Jackson et al. 1996), conflict resolution (Pawlak 1984), intelligent agents (Johnson 1998), image analysis (Mrozek and Plonka 1993), pattern recognition (Kowalczyk 1996), control theory (Mrozek and Plonka 1993), process industry (Mrozek 1992), marketing (van den Poel 1998), and so on. Rough set has become a rigorous mathematical tool, and has been implemented into software systems (Bazan et al. 1994, Øhrn and Komorowski 1997). Due to its mathematical rigor and abilities in solving practical problems, rough set theory and its applications are attracting attention from more and more domains. Due to the fact that the theory of rough set is quite mathematically involving, our introduction aims at achieving a balance between the ease of understanding by engineers and retaining its original mathematical features. As a result, the introduction will leave out any mathematical proof and simplify the description of concepts followed by an example for illustration. Also due to the length limitation and the wide scope of rough set, only those related notions are introduced. For a more detailed description of rough set, please see Ref. (Komorowski et al. 1999).

2.1 Information Systems and Decision Tables

According to the formal Rough Set theory, an *information system* is defined by a pair $S = (U, A)$, where U is a non-empty, finite set $(u_1, u_2, \ldots, u_m)$ called the *universe*; A is a non-empty, finite set of *attributes* $(a_1, a_2, \ldots, a_n)$. An attribute a maps a *universe* U into V_a for $a \in A$, where V_a is called the *value set* of a. The set $V = \bigcup_{a \in A} V_a$ is said to be the *domain* of A. Elements of U are called *objects*. For example, Table 1 excluding its last column is a very simple information system in which U has 11 objects $(u_1, u_2, \ldots, u_{11})$; A consists of two attributes (a_1 and a_2) and the A's *domain* V are the table elements (1.158705, …, -0.061620; -1.372335, …, 0.873131). Specifically for the data in Table 1, the two attributes a_1 and a_2 correspond to x_1 and x_2, respectively, for the well-known six-hump camelback (SC) function described by Eq. (1). The 11 U objects are in fact 11 random points defined by x_1 and x_2.

$$f_{sc}(x) = 4x_1^2 - 2.1x_1^4 - \tfrac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, x_{1,2} \in [-2, 2] \qquad \textbf{(1)}$$

In many applications the target of the classification is represented by an additional attribute called decision, for instance, *d* in Table 1. Information systems of this kind are called decision systems. Formally a decision system is any information system of the form $S = (U, A \cup \{d\})$, where $d \notin A$ is a decision attribute. The elements of *A* are called *conditional attributes* or simply, *conditions*. The decision system can be represented as a finite decision data table (simply called a decision table). In the decision table, the columns are labeled by conditional attributes and decision attributes; the rows are labeled by objects; and at the position corresponding to the row *u* and column *a* or *d*, the value $a(u)$ or $d(u)$ of *a* or *d* on objects from *U* appears. Table 1 including its last column is a decision system or decision table. Each row in the table describes the information about one object in *S*. Let us leave the question of how those decisions are made for the SC function to a later section, and continue our introduction of rough set by simply accepting the decision table.

Table 1 Example of an information (decision) system.

S	a_1	a_2	D
u_1	1.158705	-1.372335	1
u_2	-0.225895	-0.763139	0
u_3	0.414520	-0.553701	1
u_4	-1.080601	1.612387	1
u_5	1.831858	-0.093501	1
u_6	-0.876749	-1.934921	1
u_7	-1.972033	-1.043081	1
u_8	0.198583	1.895447	1
u_9	0.661393	-0.973502	1
u_{10}	-0.225895	-1.572636	1
u_{11}	-0.061620	0.873131	0

Let $S = (U, A \cup \{d\})$ be a decision system. With any subset of attributes $B \subseteq A$ we associate a binary relation $ind(B)$, called an *indiscernibility relation*, which is defined by $ind(B) = \{(u_i, u_j) \in U \times U$ for every $a \in B$, $a(u_i) = a(u_j)\ \}$, where $a(u_i)$ and $a(u_j)$ are values of the attribute *a* on the objects u_i and u_j from *U* respectively. If $u_i\ ind(B)\ u_j$, then we say that the objects u_i and u_j are indiscernible with respect to attributes from *B*. For example, objects u_2 and u_{10} in Table 1 having different decision attributes (0 and 1) are indiscernible from each other by the attribute a_1 because the value $a_1(u_2)$ of the attribute a_1 on the object u_2 has the same value –0.225895 as the value $a_1(u_{10})$ of the attribute a_1 on the object u_{10}. On the other hand, objects u_2 and u_{10} are discernible from each other by the attribute a_2 because the values $a_2(u_2)$ and $a_2(u_{10})$ of the attribute a_2 on the objects u_2 and u_{10} have different values 0.763139 and -1.572636 respectively. An equivalence class[1] of the B-indiscernibility relation is denoted by $[u]_B$.

In a decision system $S = (U, A \cup \{d\})$, the cardinality[2] of the image[3] $d(U) = \{k: d(u) = k$ for some $u \in U\}$ is called the *rank* of d and is denoted by $r(d)$, where d images U into the set V_d of values of the decision attribute d. We assume that the value set V_d of the decision d is equal to $\{0, 1, ..., r(d)-1\}$. Let us observe that the decision d determines a partition $\{P_0, P_1, ..., P_{r(d)-1}\}$ of the universe U, where $P_k = \{u \in U: d(u) = k\}$ for $0 \leq k \leq r(d)-1$. The set P_i is called the *i-th decision class* of S. Obviously, $r(d)$ is 2 and V_d is equal to $\{0, 1\}$ in the decision system Table 1.

2.2 Concepts of Cut and Attribute Value Discretization

2.2.1 Concepts of Cut

Let $S = (U, A \cup \{d\})$ be a decision system where $U = \{u_1, u_2, \ldots, u_m\}$ and $A = \{a_1, a_2, \ldots, a_n\}$. We assume $V_a = [l_a, r_a) \subset R$ for any $a \in A$ where R is the set of real numbers and l_a , r_a are the left and right ends, or lower and upper limits, of an attribute a respectively. P_a is a partition on V_a (for $a \in A$) into subintervals i.e. $p_a = \{[c_0^a, c_1^a), [c_1^a, c_2^a), ..., [c_{k_a}^a, c_{k_a+1}^a)\}$, for some integer k, where $l_a = c_0^a < c_1^a < c_2^a < ... < c_{k_a}^a < c_{k_a+1}^a = r_a$ and $V_a = [c_0^a, c_1^a) \cup [c_1^a, c_2^a) \cup ... \cup [c_{k_a}^a, c_{k_a+1}^a)$. Any P_a is uniquely defined by the set $C_a = \{c_1^a, c_2^a, ..., c_{k_a}^a\}$, called the set of cuts on V_a. We often connect P_a with the set of cuts on V_a defined by C_a. Then, any global family P of partitions can be represented by $P = \cup_{a \in A}\{a\} \times C_a$. Any pair $(a, c) \in P$ will be called a cut on V_a. The cut will define a new conditional attribute with binary values. For instance, with the data in Table 1, it will be shown later that the new attribute corresponding to the cut $(a_1, -0.5505)$ is equal to 0 if $a_1(u) < -0.5505$, otherwise is equal to 1. Hence, objects positioned on different sides of the straight line $a_1 = -0.5505$ are discerned by this cut or the cut discerns objects in the decision

[1]The equivalence class of an element $x \in X$ consists of all objectives $y \in X$ such that xRy, where R is called the equivalence relation which is reflexive, symmetric and transitive.

[2] Math. The property of having a certain cardinal number.

[3] Math. The element or set into which a given element or set is mapped by a particular function or transformation.

system. Then, how to construct a set of cuts with a minimal number of attribute elements that can discern all pairs of objects in the universe? This can be done using Boolean reasoning (Komorowski et al. 1999), heuristics method (Komorowski et al. 1999), Equal Width and Equal Frequency Interval Binning (Nguyen 1997), Holte's 1R Discretizer (Nguyen 1997), Statistical test methods (Nguyen 1997), Entropy methods (Nguyen 1997), etc. Here the steps of MD-heuristic method (Nguyen 1997) are introduced as following:

Step 1. Construct the information table *S** from the decision system *S*.

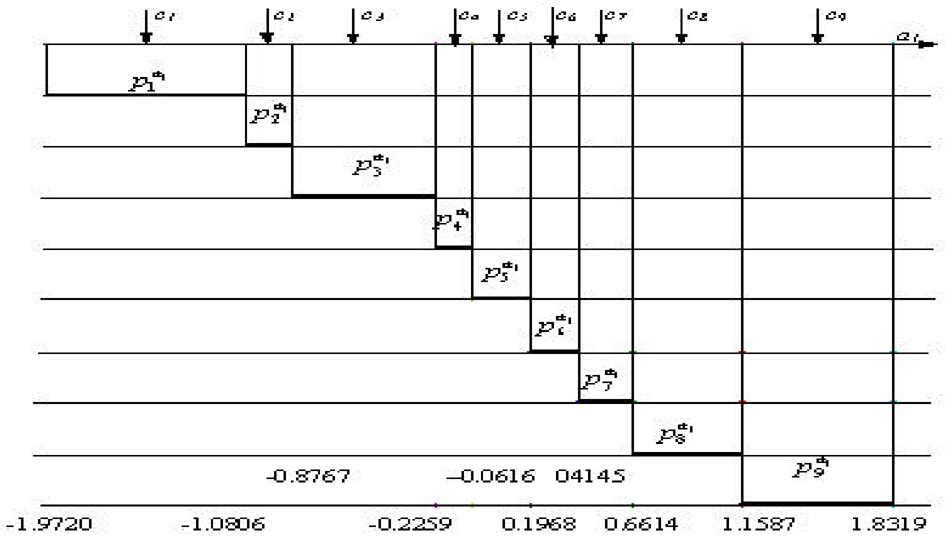
1. Introduce a Boolean variable corresponding to each attribute *a* and each interval of *a* in its information table *S*; this variable is called a propositional variable. For Table 1, a set of propositional variables is defined as

   $$VB(S) = \{p_1^{a_1}, p_2^{a_1}, p_3^{a_1}, p_4^{a_1} p_5^{a_1}, p_6^{a_1}, p_7^{a_1}, p_8^{a_1}, p_9^{a_1}, p_1^{a_2}, p_2^{a_2}, p_3^{a_2}, p_4^{a_2}, p_5^{a_2} p_6^{a_2}, p_7^{a_2}, p_8^{a_2}, p_9^{a_2}, p_{10}^{a_2}\},$$

   where $p_1^{a_1} \sim [-1.972033, -1.080601)$ of a_1, i. e., $p_1^{a_1}$ corresponds to the interval [-1.972033, -1.080601) of attribute a_1; $p_2^{a_1} \sim [-1.080601, -0.876749)$ of a_1; $p_3^{a_1} \sim [-0.876749, -0.225895)$ of a_1; …; and $p_9^{a_1} \sim [1.158705, 1.831858)$ of a_1. Similarly, $p_1^{a_2} \sim [-1.934921, -1.572636)$ of a_2; $p_2^{a_2} \sim [-1.572636, -1.372335)$ of a_2; …; and $p_{10}^{a_2} \sim [1.612387, 1.89544)$ of a_2.

   Figure 1 represents the set of cuts (a_1, c_1), (a_1, c_2), (a_1,c_3), …, (a_1, c_9) on the attribute a_1, the propositional variables $p_1^{a_1}$, $p_2^{a_2}, \cdots, p_9^{a_1}$, and the intervals corresponding to these variables. The *c* numbers are the middle point of each interval. For instance, the cut (a_1, c_5) takes the value of (a_1, *0.0685*) as c_5 *=(-0.061620+0.198583)/2*. It also can be seen from Figure 1 that each propositional variable corresponds to a cut defined within its interval. For instance, $p_3^{a_1}$ corresponds to the interval [-0.876749 -0.225895] as well as the cut (a_1, c_3).

Figure 1. The relationship among the cuts on a_1, the corresponding propositional variables and the intervals corresponding to these variables.



2. Construct the table S^* from S. As shown in Table 2, the first column lists those and only those pairs of objects with different decision values, and the first row lists all the propositional variables. For example, (u_1, u_2) is a pair listed in the table because they lead to different decision values. In contrast, (u_1, u_3) is not listed because they have the same decision value and thus need not to be discerned. The value of each propositional variable on each pair (u_i, u_j) is equal to 1 *iff* its corresponding cut (a, c) is discerning objects (u_i, u_j) (i.e. $min(a(u_i), a(u_j)) < c < max((a(u_i), a(u_j)))$ and *0* otherwise. For example, the value of $p_5^{a_1}$ in Table 2 corresponding to a cut (a_1, *0.0685*) on the pairs (u_1, u_2), is equal to *1* because this cut discerns objects (u_1, u_2) (-0.225895 < 0.0685 < 1.158705), but the value of $p_5^{a_1}$corresponding to the same cut (a_1, *0.0685*) on the pairs (u_4, u_2) is equal to *0* because this cut does not discern objects (u_4, u_2) (-1.080601 < -0.225895 < 0.0685). We can formulate this condition in another way. The value of the propositional variable P^a on the pair (u_i, u_j) is equal to 1 *iff* the interval corresponding to p^a is included in [$min(a(u_i), a(u_j))$, $max(a(u_i), a(u_j))$] and 0 otherwise. Thus, as can be seen from Figure 1, the value of $p_5^{a_1}$ on the pair (u_1, u_2) is equal to *1* because the interval corresponding to $p_5^{a_1}$ is included in [$min(a_1(u_1), a_1(u_2))$=-0.225895, $max(a_1(u_1), a_1(u_2))$=1.158705] and but the value of $p_5^{a_1}$ on the pair (u_4, u_2) is equal to *0* because the interval corresponding to $p_5^{a_1}$ is not included in [$min(a_1(u_4), a_1(u_2))$=-1.080601,

max($a_1(u_4)$, $a_1(u_2)$) =-0.225895]. The resulting new table *S** from *S* in Table 1 is shown in Table 2.

Step 2. Choose a column from *S** with the maximal number of occurrences of *1*'s. For instance, choose column $p_5^{a_1}$ because it has the maximal number of occurrences of *1*'s. If there are a few columns having the same maximal number of occurrences of *1*'s, one of the columns is randomly selected.

Step 3. Delete from *S** the chosen column and all rows that have been marked "*1*" in this column. In our example, we delete the column $p_5^{a_1}$ and the rows (u_1, u_2), (u_1, u_{11}), (u_3, u_2), (u_3, u_{11}), (u_5, u_2), (u_5, u_{11}), (u_8, u_2), (u_8, u_{11}), (u_9, u_2) and (u_9, u_{11}) because we have chosen the column $p_5^{a_1}$ in step 2 and these rows are marked 1 in the column $p_5^{a_1}$.

Step 4. If *S** is non-empty then go to Step 2; else Stop. In our example, we should go to Step 2 because the rows (u_4, u_2), (u_4, u_{11}), (u_6, u_2), (u_6, u_{11}), (u_7, u_2), (u_7, u_{11}), (u_{10}, u_2), and (u_{10}, u_{11}) remain.

Table 2 An information System S constructed from S*

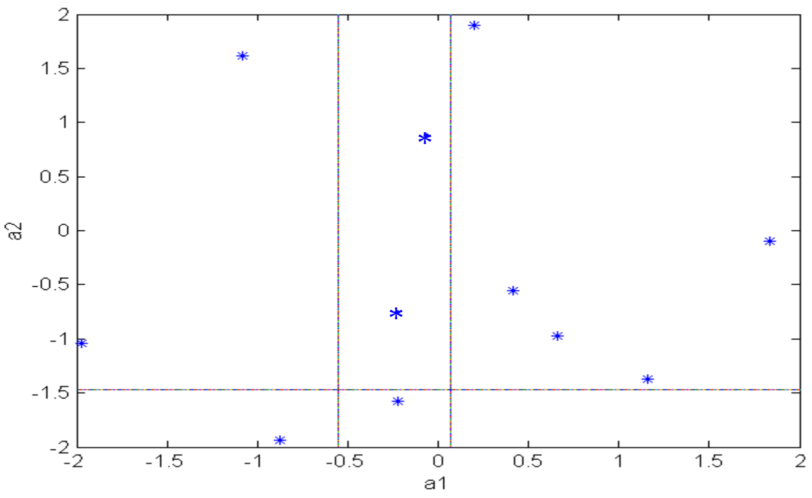
*S**	$p_1^{a_1}$	$p_2^{a_1}$	$p_3^{a_1}$	$p_4^{a_1}$	$p_5^{a_1}$	$p_6^{a_1}$	$p_7^{a_1}$	$p_8^{a_1}$	$p_9^{a_1}$	$p_1^{a_2}$	$p_2^{a_2}$	$p_3^{a_2}$	$p_4^{a_2}$	$p_5^{a_2}$	$p_6^{a_2}$	$p_7^{a_2}$	$p_8^{a_2}$	$p_9^{a_2}$	$p_{10}^{a_2}$
(u_1, u_2)	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0
(u_1, u_{11})	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0
(u_3, u_2)	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0
(u_3, u_{11})	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0
(u_4, u_2)	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
(u_4, u_{11})	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
(u_5, u_2)	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0
(u_5, u_{11})	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0
(u_6, u_2)	0	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
(u_6, u_{11})	0	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
(u_7, u_2)	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
(u_7, u_{11})	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
(u_8, u_2)	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1
(u_8, u_{11})	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(u_9, u_2)	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0
(u_9, u_{11})	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0
(u_{10}, u_2)	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
(u_{10}, u_{11})	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0

For Table 2, the algorithm first chose $p_5^{a_1}$, then $p_3^{a_1}$ and finally $p_2^{a_2}$. These three propositional variables correspond to the cuts (a_1, *0.0685*), (a_1, *-0.5505*), and (a_2, *-1.472*), respectively. The resulting set of cuts is thus P = {(a_1, *-0.5505*), (a_1, *0.0685*), (a_2, *-1.472*)} and is listed in Table 3. Same results can be obtained by applying the software RSES (Bazan et al. 1994). The geometrical representation of data's partitions and cuts is shown in Figure 2. The data in Table 1 fall in the different intervals in Figure 2. Let *n* be the number of objects and let *k* be the number of attributes of decision system *S*. The MD-heuristic method determines the best cut in *O*(*kn*) steps (Komorowski et al. 1999) . This heuristic is very efficient in terms of the time necessary for the decision rule generation and the quality of object classification.

Table 3 A set of cuts for the decision system defined in Table 1.

A	Cuts
a_1	-0.5505
	0.0685
a_2	-1.472

Figure 2 A geometrical representation of data partition and cuts.



In summary, the cutting operation generates the smallest set of attribute elements that can classify the sample points of the same decision value. Then the intervals of each attribute formed by the cutting operation are represented by integer numbers for the generation of decision rules.

2.2.2 Discretization of Attribute Values

Any set of cuts $p = \bigcup_{a \in A} \{a\} \times c_a = \{(a_1, c_1^1), ..., (a_1, c_{k_1}^1), (a_2, c_1^2), ..., (a_2, c_{k_2}^2), ...\}$ transforms from S = (U, $A \cup \{d\}$) into a new decision table $S^p = (U, S^p \cup \{d\})$, where $S^p = \{a^p: a \in A\}$ and $a^p(u) = i \Leftrightarrow$

$a(u) \in [c_i^a, c_{i+1}^a)$ for any $u \in U$ and $i \in \{0,\ldots,k_a\}$. This is to say that $a^p(u) = i$ in Table 4 represents $a(u) \in [c_i^a, c_{i+1}^a)$ in Table 1. For instance, $a_1^p(u_1) = 2$ in Table 4 represents $a_1(u_1) \in [0.0685, 2)$ in Table 1. The table S^p is called *P*-discretization of *S*. In other words, cuts make a partition of value sets of conditional attributes into intervals that are given unique integer names (Nguyen 1997). In this way the size of the value attribute sets in a decision system is reduced. For example, we use the cuts in Table 3 to discretize Table 1 into Table 4. Given the bound of a_1 [-2, 2], this set of cuts assigns the name 0 to the interval [-2, -0.5505] of a_1, the name 1 to the interval [-0.5505, 0.0685), and the name 2 to the interval [0.0685, 2]. A similar construction is done on a_2 within the bound [-2, 2] as well. The values of the new attributes a_1^p and a_2^p are shown in Table 4.

Table 4 P-discretization of the decision system Table 1 (where P = {(a1, -0.5505),(a1, 0.0685), (a2, -1.472)}).

S^p	a_1^p	a_2^p	D
u_1	2	1	1
u_2	1	1	0
u_3	2	1	1
u_4	0	1	1
u_5	2	1	1
u_6	0	0	1
u_7	0	1	1
u_8	2	1	1
u_9	2	1	1
u_{10}	1	0	1
u_{11}	1	1	0

2.3 Generation of Decision Rules

A decision system expresses all the knowledge about the model. But usually this decision system may be unnecessarily large partially because it is redundant in at least two ways. The same or indiscernible objects may be represented several times, or some of the attributes may be superfluous. The first issue can be solved by the equivalence class $[u]_B$ and the second issue by feature extraction or eliminating the superfluous attributes. After these processes, the size of the decision system can be reduced. From the last section, we can see that as the size of the attribute value sets is reduced due to discretization, the same objects are presented several times in Table 4. For instance, five rows {2 1 1} describe the same information about some objects in *S*.

Therefore, Table 4 can be reduced to Table 5 by eliminating superfluous information represented in the equivalence classes. Table 5 is then used to generate decision rules.

Before rules are generated, we briefly introduce some notions about rules. Let $S = (U, A \cup \{d\})$ be a decision table and let $V = \bigcup_{a \in A} V_a \cup V_d$. Atomic formulae[1] over $B \subseteq A \cup \{d\}$ and V are expressions of the form $a = v$; they are called *descriptors*[2] over B and V, where $a \in \boldsymbol{B}$ and $v \in V_a$. For example in Table 5, $a_1=2$ is an atomic formulae. The set $F(B, V)$ of formulae over B and V is the least set containing all atomic formulae over B and V and closed under propositional connectives: $\neg$ (negation), $\vee$ (disjunction) and $\wedge$ (conjunction). Let $\tau \in F(B, V)$. $\|t\|_S$ denotes the meaning of τ in the decision table S which is the set of all objects in U with the property τ. $\|t\|_S$ (or in short $\|t\|$) of the formulae τ in S is defined inductively as follows:

If τ is of form $\alpha = \nu$, then

$$\|t\| = \|a = v\| = \{u \in U \colon a(u) = v\} \text{ for } a \in B \text{ and } v \in V_a;$$

$$\|t \vee t'\| = \|t\| \cup \|t'\|;$$

$$\|t \wedge t'\| = \|t\| \cap \|t'\|;$$

$$\|\neg t\| = U - \|t\|.$$

The set $F(B,V)$ is called the set of *conditional formulae* of a decision table S.

A *decision rule* in S is any expression of the form $t \Rightarrow d = v$, where $t \in F(B, V)$, $v \in V_d$ and $\|t\| \neq \varnothing$. The decision rule $t \Rightarrow d = v$ is *true* in S if and only if $\|t\| \subseteq \|d = v\|$. Formulae τ and $d = \nu$ are referred as the predecessor and the successor of the decision rule $t \Rightarrow d = v$. $\|t\|$ is the set of objects *matching* the decision rule. $\|t\| \cap \|d = v\|$ is the set of objects *supporting* the rule. The decision rule $t \Rightarrow d = v$ is *minimal* in S if and only if it includes a minimal number of descriptors on its left-hand side. Assuming our decision table is consistent, we can obtain the decision rules with a minimal number of descriptors on the left-hand side. There are different algorithms to generate rules. Readers can refer to rough sets references (Pawlak 1982,

[1] A undisjoined rule or principle expressed in algebraic symbols
[2] An expression or sentence-element that has the function of describing

Komorowski et al. 1999). Decision rules listed in Table 6 are deducted from Table 5 with the RSES program (Bazan et al. 1994). For example, (a_1= 1) & (a_2 = 1) ⇒ (d = 0) means whenever the attribute a_1= 1 and a_2 = 1, the decision will be zero.

Table 5 A simplified decision system.

S^*	a_1	a_2	d
u_1	2	1	1
u_2	1	1	0
u_3	0	1	1
u_4	0	0	1
u_5	1	0	1

Table 6. Rules for the decision system

(a_1 = 2) ⇒ (d = 1)
(a_1 = 0) ⇒ (d = 1)
(a_1= 1) & (a_2 = 1) ⇒ (d = 0)
(a_2 = 0) ⇒ (d = 1)

2.4 Extraction of Attractive Regions

The main goal of the rough set analysis is to synthesize approximation of concepts from the acquired data. The set of rules from a decision system describes or reflects the nature of a decision system. The combination between the set of cuts and the set of decision rules roughly outlines the features of the decision system. When a set of cuts has been constructed and a set of rules has been induced from a decision system, we can make use of their combinations to extract the intervals that are attractive to us. For example, the rule "(a_1= 1) & (a_2 = 1) ⇒ (d = 0)" tells us that the region defined by a_1=[-0.5505, 0.0685] and a_2=[-1.472, 2] is attractive if the decision value 0 indicates a desired product performance. Assuming there is enough information to justify the decision rules, one can then search the optimum in the small region and need not to search the rest of design space. Optimization time can thus be saved.

3. Proposed Approach

According to the rough set theory, we can obtain useful rules by classifying and analyzing the information systems, represented by many data points. For computationally intensive design problems, if we can obtain such useful rules through sampling, then the design space can be reduced to smaller regions, and effort of searching for the optimal solution(s) will be significantly reduced. Before we can apply the rough set in space exploration and optimization, three questions should be addressed:

1.How to fit an optimization problem to the rough set framework?

2.How many sample points would be adequate to reflect the behavior of an objective function? Since the number of samples is problem dependent, the common strategy is to sample sequentially. This strategy leads to the third question as follows.

3. What is the criterion to terminate sampling?

This section will discuss the proposed solutions to these three questions.

3.1 Decision Threshold

For the optimization problem defined by Eq. (1), if 11 sampling points are generated, their function values can be obtained by calling the function. All the data are listed in Table 7.

Table 7 An information system for the six-hump camelback function.

Sampling No.	x_1	x_2	*f(x)*
1	1.158705	-1.372335	7.455687
2	-0.225895	-0.763139	-0.601775
3	0.414520	-0.553701	-0.452884
4	-1.080601	1.612387	17.232299
5	1.831858	-0.093501	2.165213
6	-0.876749	-1.934921	44.773839
7	-1.972033	-1.043081	5.840888
8	0.198583	1.895447	37.790520
9	0.661393	-0.973502	0.533713
10	-0.225895	-1.572636	15.127800
11	-0.061620	0.873131	-0.763318

In the optimal design domain, given a x we obtain $f(x)$, where x is a vector (x_1, x_2, …, x_m) and $f(x)$ is an objective function. In the rough set domain, we consider that x (x_1, x_2,…, x_m) as a non-empty, finite set of attributes A. Sampling points or samples are the objects of the non-empty, finite set universe U. By applying the rough set theory and considering Table 7 represents a decision system, we can see that the attribute $f(x)$ has 11 different values, i.e. $r(d) = d(U) =11$. As the number of samples or elements of U increases, the value set of the decision attribute increases. If we directly use the function value as the decision attribute, there will be nearly no pair of objects leads to the same decision and no useful rules can be generated. Therefore, a certain type of bracketing is needed for the function values. One could define as many intervals of function values as desired to see the distribution of points in each level of function values.

This is desired for design visualization or other space exploration applications. For the purpose of optimization, the division of the function values to two groups is found adequate. Hence, we introduce the decision threshold concept as following.

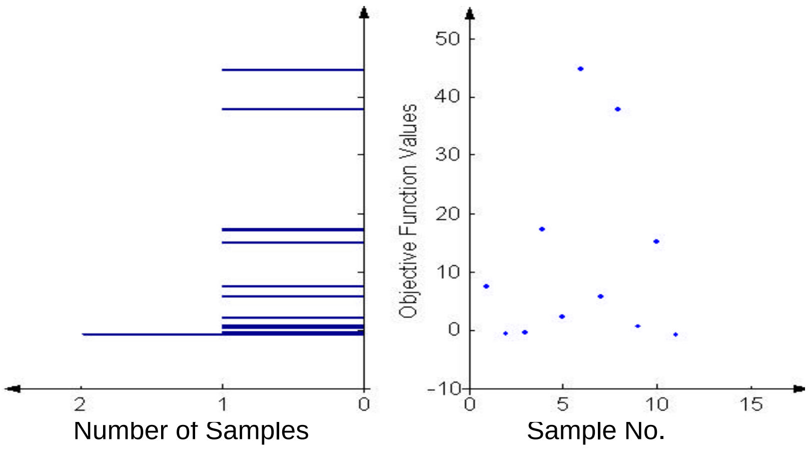
A decision threshold is a real number d_t $\{d_t$: $Min\ f(X) \leq d_t < Max\ f(X)$ | " $x \in X$, $f(x) \in f(X)$, $d_t(f(x)) = 0$ *iff* $f(x) \leq d_t$ and $d_t(f(x)) = 1$ *iff* $f(x) > d_t\}$, where $f(X)$ is a non-empty, finite set of samples for an objective function $f(x)$; $Min\ f(X)$ and $Max\ f(X)$ are the minimum and the maximum in the sampling set $f(X)$ respectively. A decision threshold d_t makes the sampling set $f(X)$ of objective function $f(x)$ be classified into two subsets $[0]_B$ and $[1]_B$ ($f(X) = [0]_B$ È $[1]_B$), $[1]_B$ is the set that the objective function value of the samples is greater than d_t; $[0]_B$ is the set that the objective function value of the samples is less than or equal to d_t. For an objective function $f(x)$, we can have different decision thresholds. Different decision thresholds possibly lead to different attractive intervals. Therefore, the efficiency of the optimization is dependent on the selection of the decision threshold d_t. On the other hand, by supplying a different d_t, one can find its corresponding sub-spaces within which all of the points will have the objective function value lower than or equal to the threshold. If many thresholds are chosen, one can basically capture the contour of the objective function. Here we will focus on the optimization and introduce an intuitive method to select the decision threshold.

A decision threshold is chosen on the basis of the current sample distribution of the objective function $f(x)$. Figure 3*b* is the function value distribution for the information system in Table 7. Figure 3*a* shows the histogram of the number of occurrence of these function values. These two figures are applicable regardless of the number of design variables. For the random sampling, the distribution of function values indicates the probability of reaching certain function values. For the data shown in Figure 3*a*, we can easily see that there are more points having a function value between -1 and 5 than any other intervals of the same length. Therefore, if the engineer is interested in finding a robust design solution, the decision threshold can be chosen as one that has the most occurrences while its value still indicates physically satisfactory performance. For example, the decision threshold d_t can be chosen as *-0.5* for the information system in Table 7. If the goal is to find the global optimum, the threshold value can be lower. The choice of the decision threshold also depends on the specific performance requirement. For example, if the design objective is cost, the design engineer usually knows about the maximum allowed cost to be competitive through benchmarking. In this case, such knowledge can be applied in the selection of d_t. If there is no *a priori* knowledge and the goal is to identify the global optimum, usually we select one value that is a little bit greater than the minimum of all sampling values as the decision threshold d_t. Thus the two figures in Figure 3 can function as a visual aid to help engineers decide on the threshold.

Once the decision threshold is chosen, samples having the function value greater than the decision threshold d_t is assigned a decision $d = 1$, otherwise, $d = 0$. Therefore, we classify all samples into two classes (*1* and *0*). Please refer to the decision system in Table 1, which is obtained from the information system in Table 7 by setting d_t=*-0.5*.

By the introduction of a decision threshold, an optimization problem can be transformed as a decision system and thus the rough set can be applied to seek the rules inherent in the samples.

Figure 3 A visual aid for selecting the decision threshold.



a. Function value distribution histogram b. Function value distribution

3.2 Sampling

For the purpose of reducing the number of samples (or, expensive function evaluations) in design optimization, a small sample set is generated first and the rough set is applied. But the rough set was originally developed for a large amount of data. Such a small set will not be able to capture the real attractive design region. Therefore, the sequential sampling is applied. In this work, the inherited Latin Hypercube Sampling method (Wang 2002, Wang and Simpson 2002) is used to sequentially sample the six-hump camelback objective function $f(x)$ and set up an information system which has a form $S = (U, x \cup \{f(x)\})$ as in Table 7. This method gradually adds samples to a given design space, and the combination of newly added points with existing points forms a new Latin Hypercube sample set with small redundancy for some variable intervals. For constrained optimization problems, the samples will be first evaluated with those constraints. If any of the constraints is violated, the point is deleted from the sample set and will not be evaluated with the objective function. The use of sequential sampling strategy leads to the question of when to terminate the sampling process.

3.3 Termination Criterion for Sampling

Definitions of Various Types of Design Spaces

Before the introduction of the termination criterion for sequential sampling, some concepts have to be defined or clarified. Given the design variable x ($x_1, x_2, \ldots, x_m$) and its value range X, the objective function $f(x)$ ($f_1(x), f_2(x), \ldots, f_n(x)$) and its local optima $f_{min}(x)$, the constraint function g(x) ($g_1(x), g_2(x), \ldots, g_k(x)$), and the decision threshold d_t, we introduce a number of definitions.

- A *design space* $S_d = (x, f(x)) = \{X \in \mathrm{R} \mid \forall\, x \in X, f(x) \in R\}$ is the value range of the optimal variable x which makes the objective function $f(x)$ sensible.
- A *feasible space* $S_f = (x, f(x), g(x)) = \{X \in R \mid \forall\, x \in X,\ f(x) \in \mathrm{R} \text{ and } \mathrm{g(x)} \leq 0\}$ is the value range of the optimal variable x which makes the objective function $f(x)$ sensible and satisfies the constraints $g(x) \leq 0$.
- An *attractive space* $S_a = (x, f(x), g(x), d_t) = \{X \in R \mid \forall\, x \in X, f(x) \leq d_t \text{ and } g(x) \leq 0\}$ is the value range of the optimal variable x within which any x makes the value of the objective function $f(x)$ equal to or less than a given decision threshold d_t and satisfies the constraints $g(x) \leq 0$.
- An *optimal design space* $S_o = (x, f(x), g(x)) = \{X \subseteq S_f \mid \exists\, x \in X, f(x) = f_{min}(x)\}$ is the set of ideal design spaces in which the local optima exists.

There exists the following relationship between these spaces:

$$S_d \supseteq S_f \supseteq S_o$$

$$S_d \supseteq S_f \supseteq S_a$$

In the previous section, we have described how we can get attractive intervals and all of the attractive intervals constitute an attractive spaces S_a. This attractive space possibly consists of a number of single or multiple, continuous or discontinuous sub-attractive spaces $[S_{a_1}, S_{a_2} ..., S_{a_p}]$. For the decision system in Table 1, we have applied the RSES (Bazan et al. 1994) to get the cuts as in Table 3 and rules in Table 6. From Table 6 or Figure 2, we know that the interval a_1 = [-0.5505 0.0685] and a_2 = [-1.472 2] forms one and only one attractive space S_a.

3.4 Overlap Coefficient and Criterion for Convergence

Given the definition of the attractive space S_a and optimal design space S_o, we hope that $S_a = S_o$ or S_a overlaps S_o. But due to the limitation of sampling, usually it is difficult to get $S_a = S_o$. As the number of sample points increases, S_a should approach to S_o. If the number of sample points in the decision system is not enough to represent the features of the objective function $f(x)$, we should continue to sample new points to be added to the information system. The next natural question is when to stop sampling? We introduce the overlap coefficient definition and a criterion for convergence.

Assuming that $S_{a_i} = \{s_{i_1} \cup s_{i_2} \cup ... \cup s_{i_m}\}$ and $S_{a_{(i+1)}} = \{s_{(i+1)_1} \cup s_{(i+1)_2} \cup ... \cup s_{(i+1)_n}\}$ are the attractive design spaces obtained through the ith and $(i+1)$th samplings respectively, we define an overlap coefficient C as following:

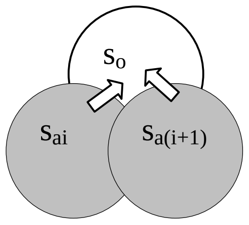
$$C = \frac{S_{a_i} \cap S_{a_{(i+1)}}}{S_{a_i} \cup S_{a_{(i+1)}}} \qquad \textbf{(2)}$$

For an engineering problem, an optimal design space S_o always exists in the feasible design space S_f. This optimal design space S_o only depends on the nature of the engineering problem itself. In order to get this optimal design space S_o, we can sample in the feasible design space S_f, then classify the feasible design space S_f into two classes of subspaces by the decision threshold d_t. One subspace is the attractive space S_a and the other is the unattractive space, $S_f - S_a$. As the number of samples increases, the overlap coefficient C of the two attractive design spaces S_{a_i} and $S_{a_{(i+1)}}$ should increase. When the number of samples or objects increases to the infinity, the two attractive design spaces S_{a_i} and $S_{a_{(i+1)}}$ will overlap each other, that is, we have

$$C = \lim_{i \to \infty} \frac{S_{a_i} \cap S_{a_{(i+1)}}}{S_{a_i} \cup S_{a_{(i+1)}}} = 1 \qquad \textbf{(3)}$$

Its geometrical explanation is as in Figure 4. When C = 1, we have $S_{a_i} = S_{a_{(i+1)}}$. Then we can consider the attractive space $S_{a_{(i+1)}}$ as the optimal design space S_o.

Figure 4 A geometrical illustration of the spaces S_a and S_o.



As design spaces and regions are defined by the variables ranges, and each variable range can be thought as an interval, the computation of the coefficient lends itself well for the interval arithmetic operations. In this work, an interval arithmetic tool called "b4m" is used (Zemke 1998). In engineering practice, we cannot sample unlimitedly, and thus it is difficult to locate the exact optimal design space S_o. From our experience, when $C \geq 0.65$, the optimal design S_o can be sufficiently approximated by the attractive space S_a.

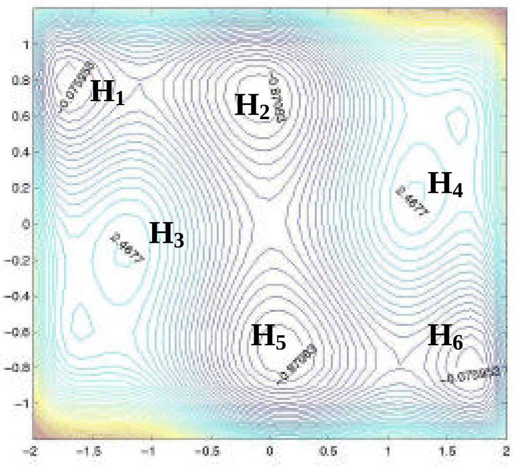
3.5 Optimization

In the attractive space S_a, it is easy to find the local optima using existing optimization methods, for example, gradient-based methods. After we get the local optima, we compare them and select one of them as a solution according to design requirements.

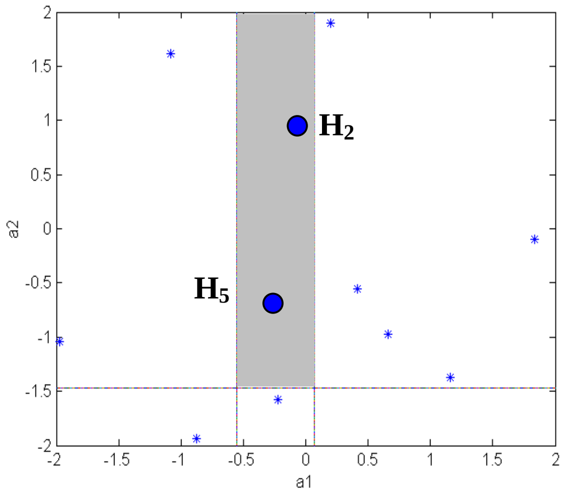
3.6 An Example

We mentioned earlier that the example data are based on the six-hump camel back (SC) function. Figure 5*a* shows the contour plot of this function. The two global optima are in the optimal space H_2 and H_5, while H_1, H_3, H_4 and H_6 indicate four local optimal spaces. Figure 5*b* shows the space partition when $d_t = -0.5$; Figure 5*c* shows the space partition when $d_t = 0$. Comparing Figure 5*b* with Figure 5*a*, we can see that Figure 5*b* captures the H_2 and H_5 of the SC function, where the big round dots are the sample points having function values lower than d_t. The convex areas H_1, H_3, H_4 and H_6 disappear in Figure 5*b* because they are classified into spaces unattractive to us after selecting d_t = -0.5. When we select d_t = 0, the convex areas H_1, and H_6 should appear in Figure 5*c*. However only H_6 appears due to the lack of sampling points in the area of H_1 (It is a very sparse sample set with only 11 points). This situation can be overcome by adding new samples or objects into the information system.

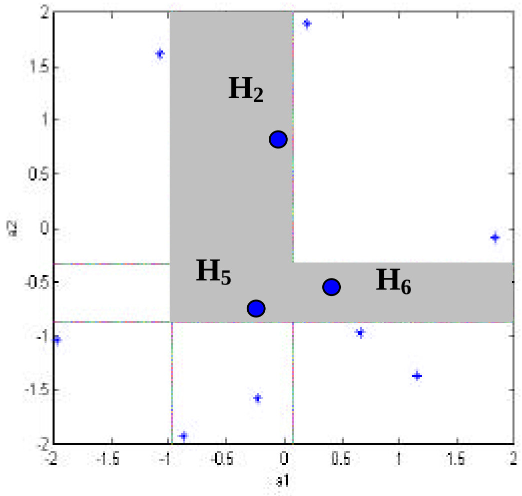
Figure 5 Comparison of space partitions with the contour of the SC function.

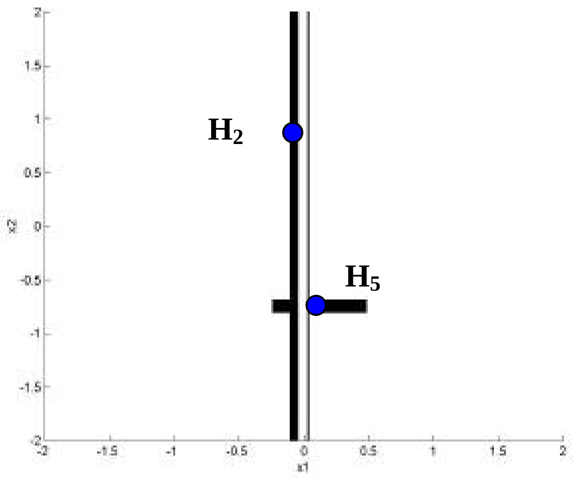


a. Contour of the original SC function



b. A representation of space partition when d_t = -0.5





c. A representation of space partition when d_t = 0 d. The final converged space with d_t = -0.5

From Figure 5*a*-*c*, one can see even with 11 samples, the entire design space can be reduced to small regions that contain the optimal spaces of SC. The decision threshold affects the number of local optima to be found and also the efficiency of the global optimization. As more samples are added and the S_a's are obtained, the overlapping criterion C will be larger than 0.65 and the final S_o can be fairly well approximated. Figure 5*d* shows the final converged region S_a with the two obtained global optima. Detailed optimization results of SC and a number of other test problems will be reported later.

4. Test of the Approach

The proposed method has been tested with a number of widely accepted test problems for global optimization algorithms. The results and comparison with other optimization methods are given as following.

4.1 Test Problems

The test problems are listed below where *n* represents the number of variables.

1. Six-hump camelback function (SC), *n=2*, as defined in Eq. (1).
2. Geometric container function (GC), *n=3*.

$$\text{Minimize } f_{GC}(x) = 0.2/x_1x_2x_3 + 4/x_1 + 3/x_3, x_i \in [0, 5]$$
$$\text{Subject to } \quad g_1(x) = -x_1 \le 0$$
$$g_2(x) = -x_2 \le 0$$
$$g_3(x) = -x_3 \le 0$$
$$g_4(x) = 2x_1x_3 + x_1x_2 - 10 \le 0$$

3. Hartman function (HN), *n=6*.

$$f_{HN}(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2], x_i \in [0, 1], i = 1, \dots, n$$

where,

i	$a_{ij}, j = 1, \dots, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	.05	10	0.1	14	3.2

i	$p_{ij}, j = 1, \dots, 6$					
1	.1312	.1696	.5569	.0124	.8283	.5886
2	.2329	.4135	.8307	.3736	.1004	.9991
3	.2348	.1451	.3522	.2883	.3047	.6650
4	.4047	.8828	.8732	.5743	.1091	.0381

4. The fourth problem involves the design of a sandwich beam (SB) developed by Messac (1996). The task is to design the sandwich beam shown in Figure 6 to support a vibrating motor. The beam consists of three layers of different materials: the mass density (ρ_i), Young's Modulus (E_i), and cost per unit volume (C_i) for each of the three material types are provided in Table 8.

Figure 6 The sandwich beam design problem.

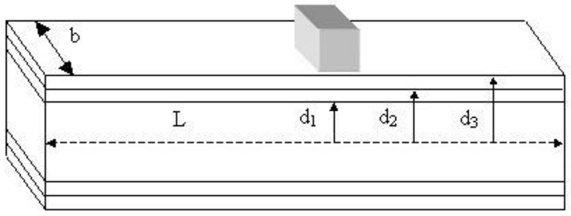


Table 8 Material properties of beam layers.

Material Type	ρ (Kg/m^3)	E (N/m^2)	C (\$/m^3)
1	100	1.6 * 10^9	500
2	2770	70 * 10^9	1500
3	7780	200 * 10^9	800

The design objective is to minimize the cost while satisfying fundamental frequency and other constraints. The cost function, f_c, and frequency function, f_f, are described as follows:

$$f_c = 2bL[c_1 d_1 + c_2(d_2 - d_1) + c_3(d_3 - d_2)]$$

$$f_f = \frac{p}{2L^2}\sqrt{\frac{EI}{m}}$$

where:

$$EI = \frac{2b}{3}\left[E_1 d_1^3 + E_2(d_2^3 - d_1^3) + E_3(d_3^3 - d_2^3)\right]$$

$$m = 2b\left[r_1 d_1 + r_2(d_2 - d_1) + r_3(d_3 - d_2)\right]$$

Several geometric constraints are imposed: an upper bound for the total mass of the beam; minimum thickness for layers two and three; and ranges for each geometric parameters, d_1, d_2, d_3, b, and L. The particular problem instantiation used for this study is:

$$\min f_c = 2x_4 x_5(c_1 x_1 + c_2 x_2 + c_3 x_3)$$

Subject to

$$f_f = \frac{p}{2x_5^2}\sqrt{\frac{EI}{m}} \geq 150,\ \ EI = \frac{2x_4}{3}[(E_1 - E_2)x_1^3 + (E_2 - E_3)(x_2 + x_1)^3 + E_3(x_1 + x_2 + x_3)^3]$$

$$m = 2x_4(r_1 x_1 + r_2 x_2 + r_3 x_3)$$

$$x_{2,3} \geq 0.01$$

$$m x_5 \leq 2700$$

$$x_1 \in [0\ 1],\ x_2 \in [0\ 0.05],\ x_3 \in [0\ 0.05],\ x_4 \in [0.5\ 2],\ x_5 \in [3\ 10]$$

where x_1: d_1, x_2: (d_2-d_1), x_3: (d_3-d_2), x_4: b and x_5: L. Based on the above equation, the initial design space is: x_1: [0 1] x_2: [0 0.05] x_3: [0 0.05] x_4: [0.5 2] x_5: [3 10].

4.2 Test Results

Table 9 lists the detail results of the test problems. Table 10 compares the results obtained using the simulated annealing (SA) method (Kirkpatrick et al. 1983), the Boender-Timmer-Rinnoy-Kan (BTRK) clustering algorithm (Boender et al. 1982, Csendes 1985), and the DIRECT method (Jones et al. 1993, Gablonsky 1998). These three methods are well known methods dealing with black-box problems with no gradient information required. The SA process is a stochastic optimization method analogous to the physical annealing of a solid. The BTRK clustering method was tested against other stochastic optimization algorithms including the SA, and the BTRK method was the winner on 45 standard testing problems of dimensions 2-30 (Neumaier 2002).

In Table 9 for the four test problems, the second column lists the number of variables; the 3nd~6th columns list, respectively, the decision value d_t, the overlap coefficient C, the number of obtained S_a's, and the number of function evaluations used in identifying the S_a's by using the proposed method. One can see that all the C's are larger than 0.65. The column labeled by "# of local min." gives the number of the analytical optima whose function values are also lower than d_t, as well as the number of optima found by further performing local optimization in captured S_a's. One can see that the proposed method can find all the local optima for each of the four test problems. The number of function evaluations needed in identifying these S_a's is limited as shown in the 6th column.

Table 9 Summary of space identification results by using the proposed method.

Func.	# of var.	d_t	C	# of S_a's	# of func. eval. to find S_a	# of local min.		Global optimum	
						Exist	Got	Analytical	Proposed Method
SC	2	-0.5	0.78	4	207	2	2	(-0.0898, 0.7127) F=-1.0316 (0.0898, -0.7127) F=-1.0316	(-0.0898, 0.7127) F=-1.0316 (0.0898, -0.7127) F=-1.0316
GC	3	4.0	0.95	1	52	1	1	(2.3798, 0.3162, 1.9429) F=3.362	(2.3798, 0.3162, 1.9429) F=3.362
HN	6	-0.9	0.72	20	297	3	3	(0.4047, 0.8824, 0.8462, 0.5740, 0.1388,0.0385) F=-3.203 (0.2017, 0.1500, 0.4769, 0.2753, 0.3116, 0.6573) F=-3.322 (0.4046, 0.8823, 0.8537, 0.5739, 0.2262, 0.0387) F=-3.203	(0.4047, 0.8824, 0.8462, 0.5740, 0.1388,0.0385) F=-3.203 (0.2017, 0.1500, 0.4769, 0.2753, 0.3116, 0.6573) F=-3.322 (0.4046, 0.8823, 0.8537, 0.5739, 0.2262, 0.0387) F=-3.203
BEAM	5	1700	0.77	7	1037	1	1	(0.1678, 0.0100, 0.0100, 0.5000, 3.0000) F=320.770	(0.1678, 0.0100, 0.0100, 0.5000, 3.0000) F=320.770

Table 10 Comparison of the optimization results with other three strategies.

Func.	Anal. Solu.	Direct			Simulated annealing			BTRK Clustering			Proposed method		
		# of func. eval.	# of optima found.	Optim.	# of func. eval.	# of optima found.	Optim.	# of func. eval.	# of optima found.	Optim.	# of func. eval.	# of optima found.	Optim.
SC	-1.032	165	1	-1.032	11128	1	-1.032	6710	1	-1.032	311	2	-1.032 -1.032
HN	-3.322	393	1	-3.321	11081	1	-0.161	9180	1	-3.320	2424	3	-3.203 -3.322 -3.203
GC	3.362	913	1	3.378	55130	1	3.362	8300	1	3.66	78	1	3.362
BEAM	320.770	433	1	605.140	55269	1	321.040	21510	1	2.69e+9	1212	1	320.772

As described before, the efficiency of the optimization depends on the chosen decision threshold. Based on the d_t's listed in Table 9, the overall number of function evaluations and the global optima are listed in Table 10, compared with results obtained by using other popular global optimization methods. As shown in Table 10, all of the four methods, including the proposed method, have found the global or close-to-global optimum for the first three test problems. The Direct and the BTRK Clustering method failed for the Beam problem. Among all four methods, only the proposed method has found all of the exact analytical solution. Also, the three chosen methods only found one global optimum; only the proposed method found all of the global optimum and those local optima that are close to the global optimum. The number of function evaluations used by the proposed method consists of the function evaluations used in identifying the attractive spaces, and the ones used in performing local optimization in all of the identified S_a's. The total number of function evaluations that the proposed method takes is much lower than that demanded by the BTRK Clustering and the SA methods, and comparable with the DIRECT method. Moreover, the decision threshold chosen for the comparison could be further reduced and so would the total number function evaluations.

5. Closing Remarks

This work proposed a rough set based approach to search for attractive design spaces and the global design optimum. The test results demonstrate that this method can effectively capture all of the global optima and neighboring local optima, with a limited number of function evaluations. Instead of searching for individual points, this method can yield "small islands" in a big design space. All the points in the "island" are expected to have function values less than a given decision threshold if enough samples are given. The proposed method can be possibly used in following applications.

1. To "contour" the objective function in a *n*-D design space by using different decision thresholds in the proposed method. Thus design engineers can jump to a smaller design space for further study and exploration. Therefore, this method provides the space exploration capability. This capability is very useful in design visualization in that the performance space can be mapped to design spaces (Eddy and Lewis 2002)
2. To search for both global optimal design and, probably, robust optimal design. For example, among the final set of sub-spaces (points in which all have satisfactory function

values), one can search for “flat” subspaces to identify practical robust designs, instead of “sharp” subspaces for the global yet no robust design solution.

3. To be combined with other metamodeling based optimization methods. The proposed method can identify attractive subspaces with conservative expenses. This method can be always used as the first step to reduce the design space, and thus to cut down the computation expense for metamodeling based optimization methods.

The contributions of this work are as follows:

1. It is the first time that the Rough Set theory and tools are introduced and successfully applied to design and optimization.
2. The definition of the overlapping coefficient and its integration with the sequential iterative sampling strategy enables the use of Rough Set in identifying the subspaces and optimum with a limited number of function evaluations.
3. The use of intuitive graphs to assist the selection of threshold. The projection of function values to one axis can help the engineer visualize the probability of reaching certain function values. Such an intuitive aid may be used for robust design as well.

It is also found that the inherited Latin Hypercube Sampling method evenly distributes all the sample points to a given space. This is advantageous as it reduces the possibility of missing important function features. On the other hand, it is not efficient for high dimensional design problems. For the purpose of optimization, a large portion of the design space is unattractive and thus it is not necessary to capture the function features in those areas. Therefore, a discriminating sampling strategy may be better integrated with the proposed method in tackling high dimensional problems. In addition, it is observed for some of the problems, the number of obtained S_a’s is large, for instance, S_a’s found for the HN function is 20 with 3 local optima as shown in Table 9. Consolidation of subspaces might be beneficial.

6. Acknowledgement

Financial support from the Natural Science and Engineering Research Council (NSERC) of Canada is gratefully appreciated.

References

1. Bazan, J., Nguyen, S. H., Nguyen, H. S., Synak, P., and Wroblewski, J., 1994, *RSES – Rough Set Exploration System, version 1.1*, Copyright © 1994-2000 Logic Group, Institute of Mathematics, Warsaw University, Poland.
2. Boender, C. G. E., Rinnooy Kan, A. H. G., Strougie, L., and Timmer, G. T., 1982, "A stochastic method for global optimization", *Mathematical Programming,* Vol. 22, pp. 125-140.
3. Csendes, T., 1985, "Two non-derivative implementations of Beender et al's global optimization method: numerical performance", Report 1985/2, Jo`zsef Attila University, Szeged, Hungary.
4. Czyzewski, A., 1997, "Learning algorithms for audio signal enhancement – Part II, Rough set method implementation for the removal of hiss," *Journal of the Audio Engineering Society* 45/11, pp. 931-943.
5. Eddy, J., and Lewis, K. E., 2002, "Multidimensional design visualization in multiobjective optimization," *Proceedings of the 9th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-2002-5621, Atlanta, Georgia.
6. Gablonsky, J., 1998, "An Implementation of the DIRECT algorithm", North Carolina State University report, Department of Mathematics, Center for Research in Scientific Computation.
7. Golan, R., and Ziarko, W., 1995, "A methodology for stock market analysis utilizing rough set theory," *Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, New York City, pp. 32-40.
8. Gu, L., 2001, "A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis," *ASME Design Engineering Technical Conferences - Design Automation Conference (DAC) (Diaz, A., ed.)*, Pittsburgh, PA, ASME, September 9-12, Paper No. DETC2001/DAC-21063.
9. Jackson, A. G., Leclair, S. R., Ohmer, M. C., Ziarko, W., Al-Kamhawi, H., 1996, "Rough sets applied to material data," *Acta Metallurgica et Materialia*, pp. 44-75.
10. Jain, P. and Agogino, A. M., 1993, "Global Optimization Using Multistart Method," Transactions of the ASME, Journal of Mechanical Design, Vol. 115, December 1993, pp. 770-775.
11. Johnson, J., 1998, "Rough mereology for industrial design," *Proceedings of the First International Conference on Rough Sets and Soft Computing*, Warszawa, Poland, Springer-Verlag, LNAI 1424, pp. 553-556.
12. Jones, D.R., Perttunen, C.D. and Stuckmann, B.E., 1993, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, October 1993, pp. 79-157.
13. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671 – 680.
14. Koch, P. N., Simpson, T. W., Allen, J. K. and Mistree, F., 1999, "Statistical Approximations for Multidisciplinary Optimization: The Problem of Size," *Special Multidisciplinary Design Optimization Issue of Journal of Aircraft*, Vol. 36, No. 1, pp. 275-286.
15. Komorowski, J., Pawlak, Z., Polkowski, L., and Skowron, A., 1999, *Rough sets: A tutorial*, in: S.K. Pal and A. Skowron (eds.), *Rough fuzzy hybridization: A new trend in decision-making*, Springer-Verlag, Singapore, pp. 3-98.
16. Kowalczyk, W., 1996, "Analyzing temporal patterns with rough sets," *Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, Verlag Mainz 1, pp. 139-143.
17. Messac, A., 1996, "Physical Programming: Effective Optimization for Computational Design," *AIAA Journal*, Vol. 34, No. 1, pp. 149-158.
18. Mrozek, A., 1985, "Information systems and control algorithms," *Bull. Polish Acad. Sci. Tech. Sci.*, 33, pp. 195-212.

19. Mrozek, A., 1992, “Rough sets in computer implementation of rule-based control of industrial processes,” In *Intelligent Decision Support – Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publishers, Dordrecht, pp. 19-31.

20. Mrozek, A., Plonka, L., 1993, “Rough sets in image analysis,” *Foundations of Computing Decision Sciences,* 18/3-4, pp. 259-273.

21. Neumaier, A., 2001, “Chapter 4: Constrained Global Optimization”, in *COCONUT Deliverable D1: Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art*, the Coconut project.

22. Neumaier, A., 2002, http://www.mat.univie.ac.at/~neum/glopt.html. Computational Mathematics group, the University of Vienna, Austria.

23. Nguyen, H. S., 1997, *Discretization of Real Value Attribute: A Boolean Reasoning Approach*, Ph. D thesis, Warsaw University.

24. Øhrn, A., Komorowski, J., 1997, “ROSETTA: A Rough Set Toolkit for Analysis of Data,” *Proc. Third International Joint Conference on Information Sciences, Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97),* Durham, NC, USA, March 1-5, Vol. 3, pp. 403-407.

25. Pawlak, Z., 1982, “Rough Sets,” *International Journal of Computer and Information Sciences* 11, pp. 341-356.

26. Pawlak, Z., 1984, “On conflicts,” *Int. J. of Man-Machine Studies*, 21, pp. 127-134.

27. Pawlak, Z., Stowinski, K., Stowinski, R., 1986, “Rough classification of patients after highly selected vagotomy for duodenal ucler,” *Journal of Man-Machine Studies* 24, pp. 413-433.

28. Reddy, S. Y., 1996, “HIDER: A Methodology for Early-Stage Exploration of Design Space,” *The proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Paper No. 96-DETC/DAC-1089, August 18-22, Irvine, California.

29. Ullman, D. G., 2002, “Toward the Ideal Mechanical Engineering Design Support System,” *Research in Engineering Design* 13, pp. 55-64.

30. Van den Poel, D., 1998, “Rough set for database marketing,” In *Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems*, Physica-Verlag, Heidelberg, pp. 324-335.

31. Wang, G. G., and Simpson, T. W., 2002, “Fuzzy Clustering Based Hierarchical Metamodeling for Design Optimization,” *Proceedings of the 9th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September, Atlanta, Georgia.

32. Wang, G. G., 2002, "Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points," *ASME Journal of Mechanical Design,* 2002, to appear.

33. Wang, G. G., Dong, Z. and Aitchison, P., "Adaptive Response Surface Method - A Global Optimization Scheme for Computation-intensive Design Problems," *Journal of Engineering Optimization,* Vol. 33, No. 6, 2001, pp. 707-734.

34. Winer, E. H. and Bloebaum, C. L., 2002a, “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. Part I: Method development,” *Structural and Multidisciplinary Optimization*, v 23, n 6, July 2002.

35. Winer, E. H. and Bloebaum, C. L., 2002b, “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. Part II: Method validation,” *Structural and Multidisciplinary Optimization*, v 23, n 6, July 2002.

36. Zemke, J., 1998, B4m: a free interval arithmetic toolbox for Matlab based on BIAS, version 1.02.004. http://www.ti3.tu-harburg.de/~zemke/b4m/index.html.