韓國經營科學會誌
第29卷 第4號
2004年 12月

인간의 학습과정 시뮬레이션에 의한 경험적 데이터를 이용한 최적화 방법

김 진 화*

An Empirical Data Driven Optimization Approach By Simulating Human Learning Processes

Jinhwa Kim*

Abstract

This study suggests a data driven optimization approach, which simulates the models of human learning processes from cognitive sciences. It shows how the human learning processes can be simulated and applied to solving combinatorial optimization problems. The main advantage of using this method is in applying it into problems, which are very difficult to simulate. "Undecidable" problems are considered as best possible application areas for this suggested approach. The concept of an "undecidable" problem is redefined. The learning models in human learning and decision-making related to combinatorial optimization in cognitive and neural sciences are designed, simulated, and implemented to solve an optimization problem. We call this approach "SLO : simulated learning for optimization." Two different versions of SLO have been designed : SLO with position & link matrix, and SLO with decomposition algorithm. The methods are tested for traveling salespersons problems to show how these approaches derive new solution empirically. The tests show that simulated learning for optimization produces new solutions with better performance empirically. Its performance, compared to other hill-climbing type methods, is relatively good.

Keyword : Undecidable Problems, Human Learning, Cognitive Science, Simulated Learning for Optimization, Combinatorial Optimization, TSP

논문접수일 : 2004년 8월 17일 논문게재확정일 : 2004년 10월 18일
* 서강대학교 경영학과

1. Introduction

This study suggests a learning approach for combinatorial optimization problems called *simulated learning for optimization.* It is based on human learning models in the cognitive sciences. The model not only solves traditional sequencing optimization problems, it also solves optimization problems inductively where only a known set of solution data obtained from observation of the real system is known.

One assumption that underlies the literature on combinatorial optimization is that *the performance of a candidate solution can be accurately measured.* Without this assumption, one cannot progress through the solution space by generating solutions in search of a good, or perhaps optimal, solution. This assumption, however, does not hold in many practical problems.

Physical implementation of this learning approach is simulated with the ideas of managing artificial memories, enforcing stimuli, learning schema, and deducing solutions from the memories. The simulated learning approach has three steps: 1) select a set of examples to be learned from available data, 2) save information from the selected examples into artificial memories, and 3) derive new solutions from the memories.

2. Optimization in Artificial Intelligence and Undecidable Problem

Problems

With many traditional algorithmic and heuristic approaches to solving business problems, there has been a gap between practice and theory, mainly due to the complexity and dynamics of today's industrial environments. Artificial intelligence techniques have given us more practical solutions for complicated problems (Kempf 1988). Learning methods, such as Inductive Learning Systems, Expert Systems, Neural Networks and Genetic Algorithms, have been successfully applied to some combinatorial optimization problems such as scheduling, sequencing, and list handling (Pinedo 1992 ; Aytug et al. 1994 ; Colorni et al. 1996). Yet the obstacle remains for traditional and artificial intelligence techniques for solving optimization problems. Can candidate solutions always be evaluated acceptably for comparative purposes? For example, what if the evaluation function of a traveling salesperson's problem is not simply travel distance, but the efficiency of travel considering the total sales and cost from the travel? From a business point of view, the cost/benefit ratio means more than travel distance alone. In academic research on algorithms and heuristics for combinatorial optimization problems, it is often assumed that a candidate solution can immediately be evaluated accurately, i.e., the performance of a candidate solution can be obtained in a decided fashion. However, as papers on "undecidability" and "uncomputability" by Alan Turing and Gödel (Stewart 1991) suggest, there are well-defined mathematical programs for which algorithms do not exist. These are called undecidable problems and their existence was proven by Turing (Turing 1936). For these problems a solution cannot be guaranteed on the basis of an unambiguous sequence of instructions, an algorithm. We generalize this notion as undecidable problems. We identify the need for additional research on systematic methods for problems characterized by high expense to accurately evaluate the quality of a candidate solution. We make this point by showing that there are important problems with this character and that the research literature has largely ignored these types of problems. We develop a foundation for defining what is meant by "high expense to accurately evaluate the quality of a candidate solution." We do this by building upon the seminal work of Alan Turing and his notion of undecidable problem. It is a paraphrasing Turing's definition : a problem is undecidable if it is impossible to accurately evaluate the quality of a candidate solution in known finite time. We adopt Turing's view, but point out that his definition excludes many important and practical problems sharing the characteristic in quotes above. Thus we expand his definition, and identify new systematic and effective methods for these types of problems. We provide practical and important real-world examples that fit our expanded definition of undecidable. For each of these examples, we explain why it does or doesn't qualify as an undecidable problem in the sense of Turing.

Many practical problems are undecidable due to the dynamics and uncertainty of systems. Chemical, metal, or food processing can be examples of problems whose exact solution quality cannot be determined easily by simulation methods (Diwekar 1977). In these cases, it is hard to predict the quality of the products before the processes are set up and tried. One must try the processes and then measure the performance of the output. Our study shows how a biologically motivated method that we term *simulated learning for optimization* can solve undecidable optimization problems. Learning models from cognitive sciences are introduced to support our construction of this simulated learning model.

2.1 The Concept of Undecidability

Despite the development of modern simulation techniques, there are still many problems that cannot be well solved with simulation. In this study, these problems are defined as *undecidable problems*, or *problems with high degree of undecidability*. The undecidability of a problem has been an issue since the 1930s (Bennet 1990). There have been complaints from industry about the gap between academic efforts and their usefulness in industry (McKay et al. 1988).

2.1.1 The Computability Theories of Alan Turing

The origin of the concept undecidability can be traced back to Alan Turing's *Turing Machine* (Alan Turing 1936). He has contributed to the development of various areas in artificial intelligence (Mackenzie and Sydney 1997). His theory is considered to have supported the invention of the first computing machine (Jones 1997). His idea of computability is that if we can clearly define procedures to solve a problem, the procedures can be achieved by computing machines. Papadimitriou and Steiglitz (1982) describe Turing's contribution to the existence of undecidable problems as below :

Are there well-defined mathematical problems for which there is no algorithm? By brilliant arguments, Turing showed that such undecidable problems do exist. A typical one is the so-called halting problem : Given a computer program with its input, will it ever halt? Turing proved that there is no algorithm that solves correctly all instances of this problem. It is possible to find some heuristic ways to detect some infinite loop patterns by examining the program and the input, but there will always be subtleties that escape our analysis. Of course, we may simply run the program and report success if we reach an end statement. Unfortunately, this scheme is not an al-

gorithm, because it is not guaranteed itself to halt!

In this study, we extend Turing definition of an undecidable problem as follows : *A problem is undecidable if it is expensive (e.g., in terms of cost and/or time) to accurately evaluate the quality of a candidate solution.* Our definition includes undecidable problems in the sense of Turing. for example, a Turing undecidable problem can be viewed as being impossible to accurately evaluate the quality of a candidate solution in finite time. In the case of the *halting problem*, for instance, it is impossible to guarantee that a computer program (i.e., a candidate solution) and associated input will halt.

2.1.2 Examples of Undecidable Problems

Following are five examples of undecidable problems. Each is a real world problem where it is difficult to accurately predict the quality or efficacy of a candidate solution without extensive testing and/or usage.

① Finding an effective algorithm for an NP-hard problem

Finding an effective algorithm for an NP-hard problem will be refered to as P. The following property puts this in perspective : "If there is a polynomial algorithm for any NP-complete problem, there are polynomial algorithms for all NP-complete problems" (Papadimitriou and Steiglitz 1982).

② Finding the best topology of neural networks

Through iterated trials of training, the optimal setting for momentum, learning ratio, number of layers, the number of nodes in each layer of a neural network is normally found.

③ Items positioning in display

Good display layouts are considered important to promote the buying motivation of customers. In practice, a person with experience and intuition decides the position of each item in a display.

④ Process sequences in chemical engineering and medical trials

Chemical reactions through a series of processes can hardly be simulated with computers. The evaluation criterion of a sequence, which is a series of processes, is the quality of the final product or the amount of materials it produces. An example can be the production of a medicine.

⑤ Advertising sequences

An issue in advertising is optimally sequencing a set of advertisements. An evaluation criterion for this sequence is total sales increase or customer evaluation of a sequence by survey.

Researchers in various academic fields such as computer sciences, industrial engineering, food science, civil engineering, and production management have suggested to us various examples of undecidable problems.

Undecidability can be understood as a matter of degree ranging between extremes. A highly undecidable problem is characterized by the lack of a generally accepted theory that can be used to predict the behavior of a candidate solution.

If there is a theory that can be used to reasonably predict the relative attractiveness of alternative candidate solutions prior to implementation, the problem is more decidable. In this study, we extend the original meaning of undecidable problems to all types of problems for which traditional simulation methods fail to find solutions.

There are three major factors in evaluating the degree of undecidability : uncertainty in the behavior of a system (Technical Feasibility), cost and time in building a simulation system (Economic Feasibility) and length of time spent evaluating a solution (Operational Feasibility). Our suggested method, simulated learning for optimization, can be applied to problems with a high degree of undecidability. We can simply call these "undecidable problems."

2.2 Experimental Design Methodologies for Undecidable Problems

As explained above, undecidable problems are widespread, important, and particularly challenging. The undecidable character of these problems is why they are sometimes addressed in an ad hoc manner through a combination of trial and error, intuition, and judgment. There is, however, a well developed, widely used and systematic approach relevant to some undecidable problems. This systematic approach is known as *experimental design (ED) methodology*. The purpose of this section is to provide a focused critical analysis of ED with respect to undecidable problems in general.

A common reason why a candidate solution for an undecidable problem is expensive to evaluate is the lack of a well-developed theory for explaining and predicting system behavior. Consequently, experiments play a large role in the investigation of undecidable problems. ED methodologies provide a framework for efficiently designing a set of experiments to be conducted (e.g., set of candidate solutions to be evaluated) and for interpreting the results. Consider the following example that illustrates an application of ED methodologies (Montgomery 1976) :

As an example of an experiment, suppose that a metallurgical engineer is interested in studying the effect of two different hardening processes, oil quenching and saltwater quenching, on an aluminum alloy. Here the objective of the experimenter is to determine the quenching solution that produces the maximum hardness for this particular alloy. The engineer decides to subject a number of alloy specimens to each quenching medium and measure the hardness of the specimens after quenching. The average hardness of the specimens treated in each quenching solution will be used to determine which solution is best.

The engineer in this example is facing the undecidable problem of determining the best quenching medium. There are two candidate solutions (i.e., oil quench and saltwater quench), and the effectiveness of each solution cannot be accurately predicted without time-consuming experimentation. ED methodologies help the engineer in this example answer such questions as the number of trials for each medium, the proper randomization of the specimens to be quenched, and the interpretation of results in the presence of white noise.

As illustrated in this simple example, ED methodologies help assess "the effect of several factors on some phenomena" (Montgomery 1976).

2.3 Learning for Combinatorial Problems in the Literature

This research is partly inspired and motivated by the ideas of heuristics derived from nature (Colorni et al. 1996). Many heuristic and non-heuristic methods have been studied to solve sequencing problems. In recent years many artificial intelligence techniques have been developed for hard sequencing problems. Colorni, Dorigo, Maffioli, Maniezzo, Righini, and Trubian provide a review in the paper, "Heuristics from Nature for Hard Combinatorial Problems" (1996). Genetic Algorithms(Holland 1975 ; Goldberg 1989), Simulated Annealing (Van Laarhoven and Aarts 1987), Sampling and Clustering (Boender et al. 1986), Tabu search (Glover 1989, 1990), Neural Networks (Hopfield and Tank 1985), and Ant System (Colorni 1991) are the well-known artificial intelligence methods for sequencing problems. Colorni and et al. (1996) classified these algorithms with four characteristics : (i) constructive vs. improving algorithms, (ii) non-structured vs. structured space, (iii) single solution vs. population of solutions, and (iv) memoryless vs. memorizing algorithms

Learning mechanisms in scheduling have been gaining importance since the 1980s. There are four categories of learning methods : rote learning, inductive learning (ID3) and neural networks learning, case-based learning and classifier systems (Aytug et al. 1994 ; Pinedo 1996). The rote learning method just memorizes the solution with good performance and uses this information without generalizing it. The inductive learning system derives rules from training examples and uses these rules to schedule with a new environment. Neural networks are also used to predict a new schedule from known examples. Case-based learning exploits experience gained from past similar problem-solving cases. Classifier systems like genetic algorithms evolve its population of solution sequences based on random crossover and mutation. Among these five methods, genetic algorithms are more often used to solve sequencing problems like job shop or traveling salesperson's problems. The other methods solve *reactive scheduling* type problems. For example, an inductive learning system for flexible manufacturing suggests the best dispatching rule among several candidates depending on the given manufacturing conditions : number of machines in the system, total buffer size, maximum relative work load, variability in machine work load, contention factor, contention factor ratio, flow allo-(Shaw, Park, and Raman 1992).

Genetic algorithms and neural networks are learning methods solving sequencing problems like job shop and traveling salesperson's problems. In genetic algorithms, learning is selecting high performance sequences from a population and reproducing new sequences into the next generation. The learning processes in neural networks consist of finding the optimal numeric representation of relationship nets by modifying weight values inside the networks through successive iterations. A so-called trained network is the final output from learning. Neural networks application to sequencing problems encounters typical optimality problems when they are applied to large size problems (Wilson and Pawley 1988). When the problem size gets large, it needs a lot of computational effort.

3. Learning Models for Undecidable Problems

This section describes human learning models in the literature. It shows how these models can explain the human learning and decision-making processes related to combinatorial optimization. The basic assumption here is that humans can induce solutions for combinatorial problems by analyzing information in known examples in a limited scale. With information and inspiration from learning processes in cognitive and neural sciences, we derive a model which simulates human learning and decision-making processes in solving combinatorial problems.

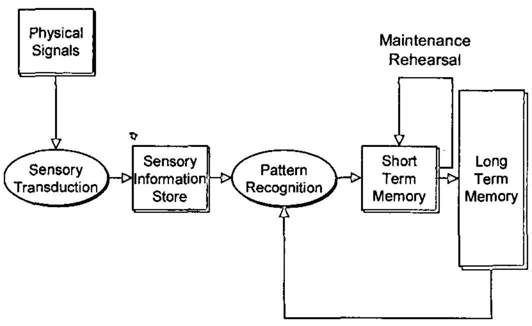
3.1 Understanding Human Learning in Cognitive & Neural Sciences

Human cognition allows the handling of a number of jobs in order : for example finding the fastest way to get somewhere though approximation, or to organize a Rubik's Cube. These are examples of sequencing experience from the past. In the case of Rubik's Cube, assume there are two competitors, one with high intelligence, but with no experience of playing the game, and the other with less intelligence but very experienced at playing the game. The experienced competitor may finish the game in a shorter amount of time because they have schemas in their memory on the best moves.

Schemas are packets of actions that may be performed in some situation to achieve an agent's goal or to take some action (Turner 1994). In sequencing cities to form an optimal round-trip tour, schemas are collections of adjacent cities in a series, as part of a whole sequence. It can also be the position of a specific city in a sequence or the link between two cities. Schemas are defined as *"mental constructs that allow patterns or configurations to be recognized as belonging to a previously learned category and which specify which moves are appropriate for that category"* (Cofer 1975). In genetic algorithms, schemas in chromosomes or sequences with good performance are produced and collected by mutations and crossover, and the schemas survive in the evolving population by natural selection. In his study of *expert-novice*, Tookey suggests that the expert has better problem-solving ability through better performance of schema acquisition (Tookey 1994). The result of their study suggests that the schemas required for efficient problem solving are acquired relatively slowly during conventional problem practice. They also suggest that problem solving skill may need many schemas, each limited to a narrow set of problems, meaning that a schema for a problem consists of many small schemas.

3.1.1 A Conceptual Learning Model in Cognitive Science

The [Figure 3.1] shows how these schemas are learned in human memory. The processing of human cognition goes through sequential stages (Norman and Bobrow 1975). Human sensory organs receive physical signals from outside and sensory memory stores the information. After detecting some patterns through the *pattern recognition* process, the signal goes through *short-term memory* and *long-term memory*.



[Figure 3.1] A View of Human Cognitive Processing

The rehearsal processes can *recycle* material in *short-term memory*. The information of a recognized pattern is evaluated for whether it should be memorized to *long term memory* or just be decayed out in *short term memory*. The *recycle* of *maintenance of rehearsal* is the repetition of sending the same information to *long-term memory* again and again to be stored. The information stored in the *long-term memory* is used in another new pattern recognition. The next two steps are big issues. How can we incorporate schemas, which lead to sequences with high performance, into memory? How can we derive sequences from these schemas? To handle the first problem, the physical representation of our memory in neurology is studied.

3.1.2 Physical Learning Models From Neural Sciences

Much about the human brain is not yet known, especially how it works on a physical level. The brain consists of small processing units called *neurons*. The *neurons* are connected in large and complex networks. The networks consist of *dendrites*, which transmit messages across various paths in the networks. The interconnected networks around a neuron are called a *synapse* or *synaptic connection*. A *neuron* has activity in the form of chemical or electrical impulses. The *synapses* bring the input to a *neuron* in the form of either a chemical or an electrical impulse. The input brought to a *neuron* either excites or inhibits the neuron's electrical or chemical activity.

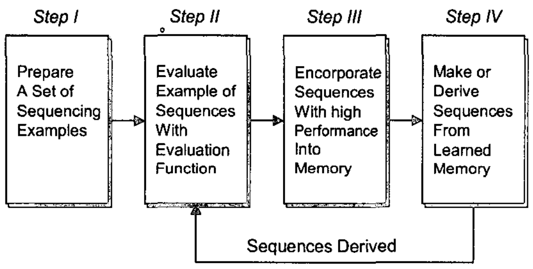
The discovery of chemical or organic substances in the human brain in the 1930s promoted the study of brain and memory, producing many clinical, neuropsychological, biophysical, and neurobiological findings about memory and learning in the brain (Brazier 1977). The components of electrical stimuli, protein synthesis, hormones, enzymes and other chemical reactions work for the processes in memory (Kumar 1980).

Capputo and Marsan (1983) show the electron micrograph of a synaptosome from the brain. It shows the synaptosomal membrane is stained dark in the cortex with strong outer stimuli. The strength and frequency of the stimuli differentiate the amount of chemicals in the related area of the brain.

In traditional neural network systems, the values are stored in the networks to represent the input and output relationship. Neural networks, however, do not perfectly simulate human learning and prediction processes. One of the problems in using neural networks is the need for updating on the basis of new information. If new information needs to be added to the existing networks, the learning processes should be repeated. This is one difference between the learning process of neural network and the learning processes of humans. The learning processes in humans are cumulative, dynamic and complex. This study uses one of these characteristics of human learning processes, that can not be found in neural networks.

3.2 Deriving a Model, Simulated Learning for Optimization

From Norman & Bobrow's model in [Figure 3.1], a conceptual model in [Figure 3.2] for *simulated learning for optimization*, which can also be called *simulated schema learning* is derived. It is simply simulating the processes of human learning and inducing sequences in their model. The first step for this model is to prepare a set of sequences. The performance of these sequences is evaluated in the next step. In job shop problems, sequences are a series of jobs, and their performances are evaluated on the basis of a merit measure such as completion time, total tardiness, or weighted tardiness. For the traveling salesperson's problems, evaluation of the sequence is a travel distance or time. The assumption here is that the sequences with high performance contain schemas, which lead to sequences with high performance. Examples with good performances are essential in deriving good sequences.


Step I
Step II
Step III
Step IV
Prepare A Set of Sequencing Examples
Evaluate Example of Sequences With Evaluation Function
Encorporate Sequences With high Performance Into Memory
Make or Derive Sequences From Learned Memory
Sequences Derived

[Figure 3.2] A Conceptual Model for Simulated Learning for Optimization

Once we memorize the good schemas in the examples, we can derive sequences with high performance. The organizational structure of the proposed approach is portrayed in [Figure 3.2].

3.2.1 Learning of Schemas

The connectivity model in *long-term memory* of humans is based on the assumption that highly inter-connected facts are stored in *long term memory* (Klimesch 1994). [Figure 3.1] shows that when sensory information is received by sensory organs, patterns or features are detected, and this information activates the related region in long-term memory (Norman and Bobrow 1975).

The electrical or chemical signal physically changes the chemical compounds in memory space in the brain. The stronger the signal, the higher the frequency becomes. The shorter the interval of frequency, the more chemical changes in the related region in the memory. In this study, two memory matrices store information from selected sequencing examples. The *position matrix* <Table 3.1> contains the position information of each element (city or job) in a sequence with high performance. The *link matrix* in <Table 3.3> stores the link information between each city or job in the sequence. It is our assumption that humans use both position and link information in high quality sequences to derive a good sequence.

We simulate these processes with two dimensional memory spaces. In the *position matrix,* the columns show the jobs or cities in sequences. The rows show positions of each job or city in sequences. The memory *M*, a matrix representation of human memory space for sequencing, has *n* columns and *n* rows. Inside the memory matrix, each cell has a value, m_{ij} (first subscript, *i* is for columns and second *j* is for rows) representing the amount of stored information from outside stimuli. The comparatively large values represent schemas in the sequences. Initially the memory matrix contains values other than all zeros.

- Stimuli Function Δ of

$$s_i = 1/(f(s_i) - average(f(s_i)))$$

The value of Δ_{ij} considers the notion that "*the stronger the stimuli is, the more electrical or chemical activity the region of memory in the brain will have*". In traveling salespersons problems, stimuli is comparative superior in travel time. The shorter the distance, the better the performance of a solution is.

3.2.2 Derivations of Solutions from Memory Matrices

Analyzing learned memory matrices to derive a desirable sequence is the next step. We first derive a sequence from a *position matrix* and improve the performance of the sequence by changing the sequence with link information in a *link matrix.*

Deriving a Sequence from Position Matrix

<Table 3.1> shows a learned memory matrix from 200 randomly generated examples. The problem here solves a traveling salesperson's problem that has 8 cities with a fixed starting city. The cities are indexed from 1 to 7, excluding the starting city. The index for the starting city is 0 and it is not shown in the memory matrix below. Since the starting city is fixed, we solve the sequence of the remaining 7 cities.

〈Table 3.1〉 An Example of Learned Position Matrix

	1	2	3	4	5	6	7
1	0	0	0	1.0	3.6	4.7	14.8
2	0	2.2	3.6	3.4	12.0	1.3	1.3
3	5.8	1.3	11.9	2.6	1.3	0	1.0
4	11.2	7.0	4.9	0	1.0	0	0
5	6.0	8.9	2.3	4.8	1.0	1.0	0
6	1.0	3.6	1.3	7.5	3.6	5.5	1.0
7	0	1.0	0	5.7	1.3	10.1	6.0

* Column : the number of index for each city
* Row : the position in the derived sequence

Deriving Good Sequences from Memory Matrices

The formal procedure for deriving solutions is given below and is parameterized by a value for *P*. With different *p* values in step 2 we normally have different sequences of solutions. The sequence with best performance is selected to be refined in the next step. As *p* gets larger to ∞, the solution from this method will become more like that obtained from the method previous introduced with <Table 3.1>.

Step 1 : Calculate weights for each city.
$w_i = (2 * i - 1) / (2 * N)(i = 1 : N)$
N : Number of cities
For the example, $w_1 = 1/14$, $w_2 = 3/14$, $w_3 = 5/14$, $w_4 = 7/14$, $w_5 = 9/14$, $w_6 = 11/14$, and $w_7 = 13/14$

Step 2 : Calculate s_j, sums of normalized values for each row(or column)with different power.

$$s_i = \sum_{1}^{N} \left(\frac{x_i^p}{\sum_{1}^{N} x_i^p} \right) W_i \quad (p = 1 : M)$$

M : integer number
If p = 2 with the problem, an example of the calculation of s_1 is below :

$s_1 = (0^2 * 1/14 + 0^2 * 3/14 + 0^2 * 5/14 + 1.0^2 * 7/14 + 3.6^2 * 9/14 + 4.7^2 * 11/14 + 14.8^2 * 13/14)$.

Step 3 : Assign numbers from 1 to N for $s_j(i = 1 : N)$ in ascending order.
We may have different sequences with different p values.

The suggested solution derived from Table 3.1 using above three steps is {0 - 4 - 5 - 3 - 6 - 2 - 7 - 1}.

3.2.3 Improving the Quality of Solution Using Link Matrix

Once a solution sequence is derived from the position matrix, the link matrix can help the solution improve the performance of the sequence by modifying its sequence with the link information. The following steps show how to derive a new solution from the position matrix and the link matrix together with a modified data set from Traveling Salespersons Problem with 10 cities. This data set is from Hopfield and Tank's paper, where they used neural networks to solve this problem (Hopfield and Tank 1985). One starting point, with coordinate (0.1, 0.5), is added to the problem, changing the TSP to 11 cities TSP with a starting city and no return. Steps of deriving solutions using the position matrix in <Table 3.2> and the link matrix in <Table 3.3> are shown next.

〈Table 3.2〉 Position Matrix for Modified Hopfield and Tank's TSP Problem

	1	2	3	4	5	6	7	8	9	10
1	64	36	78	76	32	6	19	8	12	13
2	43	59	34	46	43	27	20	19	30	22
3	36	26	31	32	39	40	38	34	27	40
4	40	25	29	33	30	47	35	34	39	32
5	33	33	28	20	29	40	32	41	34	53
6	21	31	28	17	28	42	40	46	51	39
7	21	29	29	31	15	28	53	60	35	43
8	31	29	29	16	39	35	38	36	53	38
9	28	38	26	30	34	41	42	25	37	43
10	26	37	31	43	55	38	26	40	26	20

* column : position in a sequence
* row : city index or number

〈Table 3.3〉 A Link Matrix for Modified Hopfield and Tank's TSP Problem

	1	2	3	4	5	6	7	8	9	10
0	64	36	78	76	32	6	19	8	12	13
1	0	43	31	41	48	21	36	24	40	33
2	40	0	127	29	12	13	12	18	31	24
3	39	131	0	37	12	12	14	19	16	32
4	51	16	20	0	114	26	28	8	20	17
5	32	9	10	80	0	76	40	16	11	14
6	20	10	11	23	47	0	69	78	22	25
7	28	17	15	17	30	64	0	71	30	46
8	15	11	8	9	15	72	45	0	73	55
9	20	30	28	15	20	27	42	50	0	85
10	35	40	15	17	15	25	38	51	88	0

* column : city index or number
* low : city index or number
(row 0 means link between start city to the rest of the cities)

Step 1 : Derive a solution from a learned memory matrix from 2,000 randomly generated trial solutions. About 10% of the sequences with good performance are memorized into a memory matrix. The starting point is not shown in the position matrix as it is already fixed. The derived sequence from the above position matrix in <Table 3.2> is {3 - 2 - 1 - 4 - 5 - 6 - 8 - 10 - 9 - 7} with a travel distance of 2.74582.

Step Ⅱ : Prepare link information as in <Table 3.4> from the link matrix in <Table 3.3> to change the suggested sequence found in previous step. For each row in <Table 3.4>, find the largest number, which is also a summed stimulus, and its corresponding link of two cities, row and column number where it is located in the link matrix. Give ranks to each of these rows by ascending order.

Step Ⅲ : Apply the information in <Table 3.4> to the sequence derived in Step I. By ascending order, change the sequence by applying the link information in each row one by one.

We first apply the link information in row 2 from <Table 3.4> to a given sequence solution. Row 2 has rank 1 and it suggests to link city 1 and city 5. Two possible sequences can be created using this information. Move city 5 next to city 1 or move city 1 before city 5, creating two different sequences of {3 - 2 - 1 - 5 - 4 - 6 - 8 - 10 - 9 - 7} and {3 - 2 - 4 - 1 - 5 - 6 - 8 - 10 - 9 - 7}.

If any of these newly generated sequences has better performance than the original sequence, this new sequence will replace the old solution.

As both derived sequences in this case have a performance worse than the original sequence, the old sequence still remains as the solution sequence.

When we apply the next link information of linking city 7 to city 8, we have two sequences of {3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 10 - 9} and {3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 10 - 9}. The former has a performance of 2.74582, which is better than the previous solution. The new solution is now {3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 10 - 9}, which will replace the old solution sequence. It should continue to be refined. The link information in the third ranked row makes the solution sequence {3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10} with travel distance of 2.59902.

<Table 3.4> Link Information from a Link Matrix

Rank	Value	Row	Column
4	78	0	3
1	48	1	5
10	127	2	3
11	131	3	2
9	114	4	5
6	80	5	4
5	78	6	8
2	71	7	8
3	73	8	9
7	85	9	10
8	88	10	9

After repeating this with more link information, we finally derive a sequence of {3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10} as the suggested solution sequence with travel time 2.59902. Results from experiments on more examples are discussed in section 4.

3.2.4 A Decomposition Algorithm for Simulated Learning for optimization

A decomposition approach is combined with the main idea of simulated learning for optimization to approximate solutions for a large size problem such as a TSP with a large number of cities. When a problem has a large solution space, one of the approaches can be dividing the space into sub areas and searching each sub spaces. In a combinatorial problem, such as a TSP, the sub space can be divided by each element in a sequence such as putting city1 as the first city to visit, city 2 as the first city to visit, etc.

With the decomposition algorithm, the number of elements in a solution is increasing by choosing the best city to visit next. It is a branching method by choosing the next best element into the partial solution. For a TSP, a city is selected to be the first city to visit. If the trip is round trip, you can choose any city as the first city to visit. If you choose the first city to visit, fix this city as the first element in the solution and randomly generate a number of solutions with the same city in their first position in the solution. The solutions with high performance from above random solutions are selected to store schemas into memory matrix. Analysis of these schemas suggests which city should be visited next. Now the first and second cities are fixed. Then, choose the third city to branch. These processes are repeated until all cities are visited. For a problem with large solution space, choosing one element, one city in the TSP, will reduce the solution space by 1/n, where *n* is the size of a problem. This is a divide-and-conquer approach. In each iteration, it casts a net into the solution space by generating solutions randomly and catches the best element to fill in next, the next city to visit in traveling salespersons problem.

4. Design of Tests and Analysis of Results

The primary purpose of the experiments is to assess the efficacy of simulated learning for optimization in identifying new and effective alternative solutions by understanding the "schemas" in the experiment results (in this case, experiment results refer to the costs of the many candidate solutions that were evaluated). The idea here is that SLO appears good at distilling the "schema" (and finding better solutions) of a "decidable" problem with a complex structure (i.e., NP-hard), then SLO can be effective for distilling the "schemas" from the performance of candidate solutions to undecidable problems.

4.1 Design of the Tests

Two types of simulated learning for optimization methods are designed. One is a simple simulated learning for optimization, and the other is simulated learning for optimization with decomposition algorithms. In evaluating the method of simulated learning for optimization, we compare the performance of them with that of random search, simple neighborhood search, and a genetic algorithm. We first measure the convergence time for a genetic algorithm to reach its solution. The genetic algorithm normally takes an enormous amount of time to escape local optima once its population gets homogeneous. For a traveling salespersons problem of 10 or 11 cities, a genetic algorithm takes about 1150 milliseconds to get a solution. Within this time limit for their search, the performance, which is a total travel distance, of these 5 methods is examined. Random search just randomly generates solutions in a given time and the best one of them is selected as a solution. Neighborhood search continues its search until there is no further improvement from the previous solution to the next solution.

The experiments were carried out to find the best model for simulated learning for optimization. Various **stimuli functions** such as $f(y) = 1/(1+e^{-y})$ have been evaluated to normalize the performance of the selected solutions for SLO. Tests on possible combinations of different memorizing methods and stimuli functions have been tried. When deriving solutions from a set of known solution examples, there must be reasonable number of examples. For example, for a traveling salesperson's problem with 11 cities, at least 500 sequences are needed to generate solutions with good quality. If the number of sample size is too small, it is hard to derive solutions with good quality. Different sample sizes have been tested to determine a minimum sample size for simulated learning for optimization with decomposition algorithms. The function below "# of examples needed in each iteration of $m = 500 + (m-6) * 1000$, where m decreases n to 6 (n : the number of cities in TSP ; $n > 5$)" is used to find the number of sequencing examples needed in each iteration with different number of cities for the method. The *random generation method* is selected to generate sequencing examples. Compared to *neighborhood search* and *regional best*, it supplies diverse sequencing examples from a broad solution space.

4.2 Preparing a Set of Selected Examples to Obtain Required Schemas

Three generation methods for preparing a set of sequences with high performances are introduced in this study.

Generation Method 1 : Random Generation

This method randomly generates *n* sequences and memorizes the information from the best *k* % sequences of them. The larger *n* is, the better sequences we can derive, but the more computation time is needed to induce optimal sequences. The sequences with their performance above average are used as examples for learning. The advantage of this method is its simplicity.

Generation Method 2 : Selecting the Best Neighbor in Each Step of Generic Neighborhood Search

Neighborhood search method searches its neighbor region from a random sequence as a starting point. The next search starts again from the best sequence in the neighbor region. Whenever it finds better neighbors, it chooses the best among them as next search point. The idea is to collect the best neighbor as a learning example when it searches the neighborhoods.

Generation Method 3 : Selecting Regional Best from Random Generation

N sequencing points in the solution space are produced randomly. The best neighbors for each of these *n* points are used for learning examples.

- Data Set : The number representing each data set
- RS : Random Search
- GA : Genetic Algorithm with time limit of 1150
- SLO : Simulated Learning for Optimization
- SLOD : SLO with Decomposition Algorithms
- The Best Method : The best method for a given data set

〈Table 4.1〉 The Performance Comparison of Four Methods

Data Set	RS	GA	SLO	SLOD	The Best Method
0	2.84	2.52308	2.59902	2.51257	SBLD
1	2.68	2.50374	2.46991	2.29645	SBLD
2	2.27	2.15083	2.14484	2.03925	SBLD
3	3.25285	3.49404	3.13	2.58087	SBLD
4	3.12	2.8325	3.01544	2.88512	GA
5	3.25	2.94897	3.09	2.86831	SBLD
6	2.7644	2.4616	2.4616	2.588867	GA, SBL
7	2.92417	2.56669	2.95416	2.56677	SBLD
8	3.06616	2.65667	2.6968	2.85429	GA
9	2.73579	2.37992	2.71019	2.17771	SBLD
10	3.57114	2.91886	3.16	2.88705	SBLD
11	2.65585	2.50606	2.89841	2.38122	SBLD
12	3.42686	2.93043	2.73913	2.85295	SBL
13	2.76855	2.57924	2.90974	2.4948	SBLD
14	3.06165	2.34199	2.31738	2.31735	SBL, SBLD
15	3.05218	2.55964	2.68435	2.55964	GA, SBLD
16	2.7579	2.46766	2.68435	2.45892	SBLD
17	3.245	2.87183	2.79329	2.82145	SBL
18	3.32	3.14465	3.40083	3.28838	GA
19	3.288	2.76048	3.17751	2.7474	SBLD
20	2.772	2.43752	2.75707	2.46363	GA
21	3.148	2.64028	3.19187	3.32337	GA
22	2.88795	2.67661	2.28619	2.20562	SBLD
23	2.46013	2.34123	1.93183	2.21423	SBL
24	3.13995	2.63061	2.61993	2.97254	SBL
25	3.289	2.48862	2.70783	2.48862	GA, SBLD
26	3.392	3.11134	3.32964	3.0898	SBLD
27	2.509	2.24522	2.15991	2.16373	SBL
28	2 7531	2.11891	2.69829	2.38508	GA
29	3.14	2.91991	3.71248	2.99734	GA
30	3.247	3.0706	3.15229	2.96049	SBLD

This is a hybrid method of generation method 1 and generation method 2 explained previously.

The problem tested here is a TSP with the objective function of minimizing the total travel distance. A set of TSP is randomly generated and the five different methods solve the problems in a limited time.

4.3 Results from Tests

The test results reported in <Table 4.1> show that simulated learning for optimization, with or without a decomposition algorithm, produces solutions with good performance compared to existing local search methods such as random search, neighborhood search and genetic algorithm. In modern industry, producing a solution with reasonable quality in a reasonable time is sometimes more important than producing solution with better quality in a lot more time. When there is a tight time limit on the need for a solution, this approach can be valuable as it produces solutions in a comparatively short time. If we evaluate the efficiency of methods, the ratio between the quality of solution and computing time should be considered. In many practical applications, the time needed to improve the quality of a solution by the last small percentage surpasses the benefit from the improvement. When the weight on the time factor is comparative large, the methods suggested in this study may be more efficient than other methods. In environment of modern industry, which is dynamically changing, suggesting solutions in real time is important feature. Simulated learning for optimization is also an efficient tool when there are known set of trial solutions available from past history, and there need to derive a better solution from these past trials. SLO analyzes these past trial solutions and suggest a new solution with better performance with high probability by combining schemas in them.

5. Conclusion

In this study, we have defined a problem, for which we cannot predict an output with an input unless we try the system, "an undecidable problem" or "a problem with a high degree of undecidability." This study has introduced "simulated learning for optimization" for highly undecidable problems. Examples of highly undecidable problems are sought in our real world. From the literature review on combinatorial optimization and on experimental design methodologies, we conclude that these methods in the literature are not suitable methods for the highly undecidable combinatorial problems. However, from the literature review on human learning, we concluded that humans can roughly solve a simple combinatorial problem such as a sequencing problem based on learning from known examples. The theories on schemas support our assumption that schemas in high quality solutions can be identified and combined to produce new solutions with better quality. The learning models in human learning and decision-making related to combinatorial optimization in cognitive and neural sciences were designed and simulated to solve the problem. We call the approach SLO (Simulated Learning for Optimization).

The study shows how simulated learning for optimization solves sequencing style problems by inducing solutions from memory matrices in a limited number of trials. It also shows how the approach derives solutions from a set of known solution examples. The induced solution from the method is better than the best one in the known examples with better chance. To solve combinatorial problems such as traveling salespersons problems, a modified version of simulated learning for optimization called SLOD (Simulated Learning for Optimization with a Decomposition Algorithm) is considered. This is an empirical branching method that produces a whole solution by finding the best element in a partial list as in the branch and bound method. It is better in its performance than simulated learning for optimization.

References

[1] Aytug, H., S. Bhattacharyya, G.J. Koehler and J.L. Snowdon, "A Review of Machine Learning in Scheduling," *IEEE Transaction on Engineering Management*, Vol.41, No.2(1994), pp. 165-171.

[2] Baker, K.R., *Elements of Sequencing and Scheduling*, 1995.

[3] Biegel, J.E. and J.J. Davern, "Genetic Algorithms and Job Shop Scheduling," *Computers and Interstrial Engineering*, Vol.19(1990), pp. 81-91.

[4] Bower, G.H., "Organized Memory," In Morris P.E. and M.A. Conway, *The Psychology of Memory*, Vol.I(1993).

[5] Cofer, C.N., *The Structure of Human Memory*, W. H. Freeman Company, San Francisco, California, 1975.

[6] Colorni, A., M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini and M. Trubian, " Heuristics from Nature for Hard Combinatorial Problems," *International Transactions in Operational Research*, Vol.3, No.1(1996), pp.1-21.

[7] De Robotis, E., "The Synaptosome, Tow Decades of Cell Fractionation of the Brain," *International Brain Research Organization Monograph Series, Neural Transmission, Learning, and Memory*, Vol.10(1983).

[8] Deese, J., "On the Structure of Associative Meaning," *Psychological Review*, Vol.69, No.3 (1962), pp.161-175.

[9] Dietterich, T.G. and R.S. Michalski, "Learning to Predict Sequences," In Michalski, R.S., J.G. Carbonell and T.M. Mitchell, *Machine Learning* Vol.II, Morgan Kaufmann, Los Altos, California(1986), pp.63-106.

[10] Dong, Z., *Artificial Intelligence in Optimal Design and Manufacturing*, Prentice Hall (1994), pp.153-170.

[11] Eysenck, M., W., *Human Memory : Theory Research and Individual Differences*, Pergamon Press, Elmsford, New York, 1977.

[12] Glover, F. and M. Laguna, "Target Analysis to Improve a Tabu Search Method for Machine Scheduling," *Advanced Knowledge Research Group*, US West Advanced Technologies, Boulder, Co., 1989.

[13] Glover, F., "Tabu Search", *Modern Heuristic Techniques for Combinatorial Problems*, Colin R. Reeves (Ed), Blackwell Scientific Publications, Oxford(1993), pp.70-150.

[14] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1988.

[15] Hintzman, D.L., "Schema Abstraction in a Multiple-Trace Memory Model," *Psychological Review*, Vol.93, No.4(1986), pp.411-428.

[16] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[17] Hopfield, J.J. and D.W. Tank, "Neural Computation of Decision in Optimization Problems," *Biological Cybernetics*, Vol.52(1985), pp.141-152.

[18] Jones, D.J., *Computability and Complexity From a Programming Perspective*, Massa-

chusetts Institute of Technology(1997), pp.5-245.

[19] Kempf, K., "Practical Applications of Artificial Intelligence in Manufacturing," *Artificial Intelligence in Manufacturing, Assembly and Robotics*, R. Oldenbourg Verlag München Wien(1988), pp.1-27.

[20] Kinnear, K.E., *Advances in Genetic Programming*, MIT Press, 1994.

[21] Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing," *Science* 220(1983), pp.671-680.

[22] Klatzky, R.L., *Human Memory : Structures and Processes*, W.H. Freeman and Company, San Francisco, California, 1975.

[23] Klimesch, W., *The structure of Long-Term Memory, A Connectivity Model of Semantic Processing*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1994.

[24] Lo, Z. and B. Bavarian, "Scheduling with Neural Networks for Flexible Manufacturing Systems," *Proceedings of 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California(1991), pp.818-823.

[25] Loftus, G.R. and E.F. Loftus, *Human Memory The Processing of Information*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.

[26] Lowerre, B.T., *The Harpy Speech Recognition System*, Ph.D. Thesis, Caregie Mellon University, Pittsburgh, PA, 1976.

[27] Mackenzie, A. and U. Sydney, "Undecidability : The History and Time of the Universal Turing Machine," *Configurations*(1997), pp.359-379.

[28] Malek, M., S. Guruswamy, M. Pandya and H. Owens, "Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem," *Annals of Operations Research*, 21(1989), pp.59-84.

[29] Matsuo, H., C.J. Suh and R.S. Sullivan, "A controlled search simulated annealing method for the single machine weighted tardiness problem," *Annals of Operations Research* 21(1989), pp.85-108.

[30] McKay, N.N., F.R. Safayeni and J.A. Buzacott, "Job-Shop Scheduling Theory : What Is Relevant?" *Interfaces* 18 : 4(July-August 1999), pp.84-89.

[31] Montgomery, D.C., *Design and Analysis of Experiments*, John Wiley & sons(1976), pp.341-367.

[32] Norman, D.A. and D.G. Boblow, "On the Role of Active Memory Processes in Perception and Cognition," *The Structure of Human Memory*, W.H. Freeman and Company, San Francisco, California (1975), pp.115-132.

[33] Ovacik, I.M. and R. Uzsoy, *Decomposition Methods for Complex Factory Scheduling Problems*, Kluwer Academic Publisher, 1997.

[34] Papadimitriou, C.H. and K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs(1982), pp.157.

[35] Pinedo, M., *Scheduling : Theory, Algorithms, and Systems*, Prentice Hall, New Jersey, 1996.

[36] Piramuthu, S., N. Raman, M.J. Shaw and S. C. Park, "Integration of Simulation Modeling and Inductive Learning in an Adaptive Decision Support System," *Decision Support Systems*, Vol.9, No.1(1993), pp.127-142.

[37] Rose, S., "Transient and Lasting Biochemical Response to Visual Deprivation and Experience in the Rat Visual Cortex," *International*

Brain Research Organization Monograph Series, Vol.10, *Neural Transmission, Learning, and Memory,* 1983.

[38] Schmitt, F.O., "Molecular Neurobiology in the Context of the Neuroscience," In Quarton G.C., . Melnechuk T. and Schmitte F.O., *The Neurosciences,* 1991.

[39] Shaw, M.J., S.C. Park, N. Raman, "Intelligent Scheduling with Machine Learning Capabilities : The Induction of Scheduling Knowledge," *IIE Transactions,* Vol.24, No.2 (1992), pp.156-168.

[40] Simon, H.A. and K. Kotovsky, "Human Acquisition of Concepts for Sequential Patterns," *Psychological Review,* Vol.70, No.6(1963), pp. 534-546.

[41] Turner, E.H., "A Schema-based Approach to Cooperative Problem Solving with Autonomous Underwater Vehicles," Technical Report, Department of Computer Science, University of New Hampshire, 1994.

[42] Tookey, K.R., *Arithmetic Schema and Strategy Instruction,* Ph.D. Dissertation, University of Wisconsin at Madison, 1994.

[43] Turing, A, "On Computable Numbers, with an Application to the Entscheidungs-problems," *Preceedings of the London Mathematical Society* (2), 42(1936), pp.230-265.