Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 4, pp.59－65, December 2008.

전문가시스템 구축을 위한 KACE 구조에 대한 연구

김 선 욱[†]

단국대학교 산업공학과

Architecture of Knowledge Acquisition Through Computer Experimentation (KACE) to build a Knowledge-Based Expert System

Sun-Uk Kim[†]

Department of Industrial Engineering, Dankook University

전문가시스템의 성공을 좌우하는 지식추출은 주요 애로공정 중의 하나로 알려져 있다. 설상가상으로 전문가의 부재, 새로운 또는 복잡한 문제 등 영역의 특성상 전문가시스템 개발은 실패할 수 있다. 이러한 문제점을 극복하기 위하여 본 논문에서는 KACE 구조를 제안하였다. 본 구조는 작업 발생기, 작업 실행기, 작업 평가기, 규칙 발생기와 전문가시스템 등 5개의 주요 요소로 구성되어 있다. 이 구조를 이용하여 NP-complete인 일정계획 문제에 대한 전문가시스템이 어떻게 구축될 수 있는가를 예시하였다.

Keywords : Knowledge Acquisition, Expert System, KACE, Knowledge-Based System

1. Introduction

Knowledge about the problem domain, or "knowledge acquisition," is acquired either from the study of published literature or from human experts in the domain. The expertise to be elucidated is a collection of specialized facts, procedures and judgemental rules about the narrow domain area, as opposed to general knowledge about the world. For reason of its difficult and time-consuming nature, knowledge acquisition is the major bottleneck [3, 12].

Expert system or Knowledge-Based Expert System (KBES) is a computer program that emulates the behavior of human experts who are solving real-world problems associated with a particular domain of knowledge. The key to the success of the expert system is the validity and completeness of the system's knowledge. When this knowledge is combined with various AI inferencing techniques, the outcome is a system that can solve problems and obtain results that sometimes exceed the performance of a human expert.

Since knowledge acquisition lies at the heart of the development of the KBES, the process of obtaining that knowledge is the key to building a successful KBES. As a result, most research on knowledge acquisition has stressed effective interview techniques. The transfer and transformation of knowledge required to represent expertise for a program may be automated or partially automated on some special cases. For most cases, the knowledge engineer is required to communicate with the expert and the system. The knowl-

논문접수일 : 2008년 06월 17일　　논문수정일 : 2008년 07월 03일　　게재확정일 : 2008년 07월 03일
† 교신저자 kimsunuk@dankook.ac.kr
※ The present research was conducted by the research fund of Dankook University in 2007.

edge engineer must acquire knowledge and incorporate it into the system.

Most of the current KBESs have been developed by interviewing human experts. However, the interview method does not work when the problem is new or NP complete, hard, or when acceptable expert knowledge does not exist. Even though expert knowledge does exist, in many cases it is often difficult to obtain an expert's true opinion. These problems has been pointed out in detail [2].

Even though automatic knowledge acquisition method can be applied, it is prerequisite for the problem to be defined well in terms of goal criteria and attributes. From a practical point of view, it is very hard to meet this requirement for NP-complete, hard problems like the scheduling problem for manufacturing systems [15].

As Kempf [8] has pointed out, there are few practical results in spite of much technical work done in production scheduling. Furthermore, both holding and late costs may be used as optimizing criteria. Implementing these points, however, significantly complicates the scheduling problem. Kumara et al. [10] claimed that in this situation, it is worthwhile to construct a KBES, a tool of Artificial Intelligence.

Thus, this study proposes an architecture to solve these types of problems and gives an example demonstrating it.

2. Knowledge Acquisition

2.1 Knowledge acquisition through interview

Most of the current KBESs have acquired knowledge through interview. Because most expert systems are built in cooperation with one or more human experts, the knowledge engineer must have good communication skills. In addition, the person must be thorough and well organized so that sessions with the experts are as productive as possible.

Pigford et al. [13] proposed three processes for facilitating knowledge acquisition through interview such as interaction process, knowledge-acquisition process, and interview process. The key to the knowledge-acquisition process is the interaction between the knowledge engineer and the human expert. The query/response process should be, ideally, a flow of the expert's knowledge and strategies with minimal or nonexistent gaps.

The knowledge-acquisition process usually involves more than just consulting an expert; it also may involve posing sample problems for solution and discussing their solutions. Finally, the interview is probably the technique most often used to acquire knowledge from the human expert. Good interviewing is an art, but the knowledge engineer can use some tested techniques to make the process more effective.

To summarize, the knowledge engineer must gain an understanding of the expert's thinking process. This is not an easy task, but careful questioning techniques to elicit maximum knowledge and introspective modeling can be very useful. Finally, the knowledge engineer must be continually aware of the imprecise nature of language in verbal communications.

In the direct interview method, a knowledge engineer interviews the expert and extracts the appropriate expertise. In automatic knowledge acquisition method, however, knowledge engineer is replaced by a computer program capable of generalizing specific data from which conclusions may be drawn. Hart [5] describes clear distinctions from various point of view. For example, while the former is versatile, the latter requires careful choice of examples and attributes for training set. While expert can resolve contradictions and can use fuzzy terms like "sometimes," the latter indicates problems with contradictions or gaps.

2.2 Knowledge acquisition through induction

Knowledge acquisition is the hardest task in building a KBES, and this is especially true when no expert exists. As a result, an automatic knowledge acquisition method through induction plays an important role.

From the time AI techniques were introduced, researchers have thought to understand the process of learning and the problem of creating computer programs that can learn. Cohen and Feigenbaum [4] maintained that there were four basic learning situations: rote learning, learning by being told, learning from examples, and learning by analogy. Among these options, the learning from examples is the learning situations which is most fully understood.

A program that learns form examples must acquire specific facts from some external environment. It then analyzes these facts to infer general rules, resulting in the formulation of the decision process and enabling prediction of examples not contained in the training set. This mode of learning is called inductive learning. Two of the most popular method for the inductive learning include the candidate-elimination algorithm or the version space algorithm [11], and the concept learning system (CLS) algorithm [7].

Though the candidate-elimination algorithm is very effective, it has two weakness that limit its ability to solve real problems. First, it is not well suited to learning situations in which there may be errors in the training instances. Second, it cannot learn concept descriptions that involve significant amounts of disjunction.

On the other hand, CLS is well behaved in those cases where there is noise in the data. It constructs a decision tree that attempts to minimize costs of classifying an object. ID3 [14] is a CLS refinement, which responds to the problem of the linear growth of execution time as a classification problem grows in complexity. However, from the practical point of view, several difficulties with the ID3 algorithm has been reported. The algorithm is compared with other well-known algorithms such as Rules Family, ILA, and Rex-1 [1].

The extended version of ID3 is C4.5, which consists of the following four programs: the decision tree generator, the production rule generator, the decision tree interpreter and the production rule interpreter. In brief, the program starts with a randomly-selected subset of the data (called a window), generates a trial decision tree, adds some misclassified objects, and continues until the trial decision tree correctly classifies all objects not in the window.

3. KACE Architecture

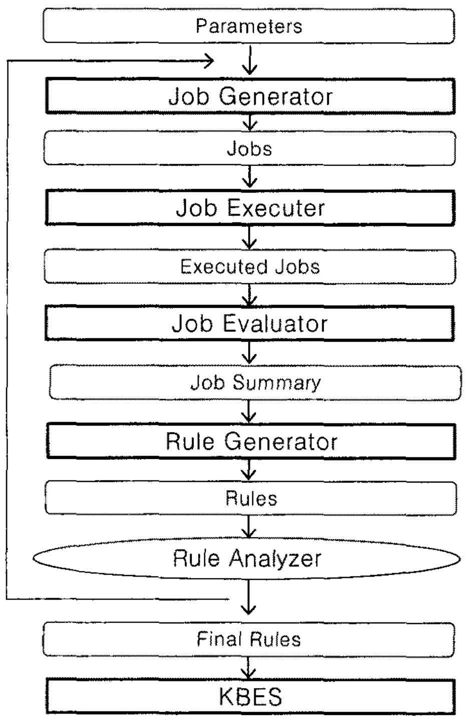
Most current expert systems were developed by interviewing human experts. As Bell [2] has pointed out, however, an expert system approach can fail because an expert is not available (even though expert knowledge does exist) or because there is no expert.

In order to solve this problem, an alternative method [9] was suggested: to train a non operator, allowing the operator to practice with a simulated system, accumulate experience, and then build an expert system using this newly acquired expertise. However, problem characteristics may limit the potential effectiveness of that approach. For example, since problem attributes have continuous rather than categorical values, and the number of attributes is large, it is almost impossible for a human operator to be trained to do well.

Furthermore, it is almost impossible to enumerate all possible cases because of system and job dependent and to decide what kind of attributes we use. Considering this situations, knowledge engineer needs some aids such as simulator, machine learning tools and so on.

Since there was no way to acquire knowledge under this situation, a method for knowledge acquisition through computer experimentation (KACE) was devised. The underlying concept of KACE was to develop heuristic rules experimentally in order to generate better solutions. A large number of problems were generated and evaluated in terms of objective criteria and attributes. The problems and their results were entered into an induction program, which was used to determine the relationships that existed between the problems and their results. This derived knowledge was then represented in the KBES.

The architecture of KACE mainly consists of five components: a job generator, a job executer, a job evaluator, a rule generator and an expert system. In <Figure 1>, a summary of the KACE procedure, rectangles indicate processors and an ellipse does manual process, while rectangles with rounded corners represent stores such as files, a database, or a temporary store.



<Figure 1> KACE Architecture

The job generator is used to provide a large number of jobs from which the induction program learn. To compare the methods defined, each of the methods are applied to each job to determine which method is best. The job evaluator examines the results generated based on the same unit of measurement. The rule generator is used to generate rules automatically from examples found in the Job Summary database. The processors, from the job generator through the rule analyzer, were repeated, based on the number of training instances, until the best set of rules were determined. In general, as the number of examples were increased, classification performance improved. Note, however, more rules do not always assure better performance.

4. An Example : Schedule-Based MRP (SBMRP)

4.1 Overview of SBMRP

Though a wealth of literature can be found in the areas of scheduling and MRP, there is sparse literature relating MRP to job shop scheduling problems. In 1982, Hastings et al. [6] developed a integrated approach, called SBMRP. It assumes a master schedule which states end product requirements by date and by quantity. The master schedule may be made up of direct customer demands, market forecasts, or a mixture of both. This master schedule is translated into a production schedule which gives sufficient information to control the production activity. Along with the production capacities, job and operation details etc., several BOM are recorded in a database suitable for a production scheduling program.

The principal method that SBMRP uses is to load the jobs forward to finite capacity, which guarantees a feasible schedule. With attention to precedence constraints, jobs are loaded one at a time, with the due dates on the batches of the products determining the order, onto the required machines. If there is no precedence relation between any of the jobs, the program could load them in any order. Actually, the program loads the jobs in the order in which they appear in the input file.

For each operation of each job, materials required are specified and the scheduling of the operations automatically creates a time phased material requirement plan. In summary, the SBMRP system produces a feasible production schedule quickly by a forward load to the finite capacity. Based on a feasible production schedule, a time phased material requirements plan is produced.

Each part and Each subassembly will have a holding cost. Furthermore, the violation of the due dates incurs a late cost for the product. Consequently, we need a schedule minimizing both the holding cost and the late cost. However, existing studies [6, 17] with SBMRP have not focused on a better schedule, which minimizing the two cost, but have focused instead only a feasible schedule satisfying demand with limited capacity. The forward method called Hastings' approach minimizes the holding cost, but it may increase shortage cost. On the other hand, the backward method that White used may increase the shortage cost.

4.2 Model Assumptions for the SBMRP

A model of a job shop was developed to provide the means for studying MRP scheduling problems. The model consists of five machining centers. Customer's orders with delivery dates specify the jobs. Each job has a unique BOM. From this BOM, the due dates for all parts to be manufactured can be derived. The operation sequence and processing times for the parts were generated randomly from uniform and exponential distributions, respectively. The BOMs in this model included at most six part types for processing.

The MRP scheduling problem addressed in this study may be defined as follows :

- The scheduling objectives are to process all jobs and to minimize the total cost, which is a sum of the late cost and holding cost, assuming that the due date has to be kept.
- Although the job is composed of distinct operations, no two operations for the same job may be processed simultaneously.
- Each operation, once started, must be completed before another operation may be stared on that machine.
- Each job must be processed to completion.
- A job represents the processing of a set of different parts.
- Late unit cost is equal to holding unit cost.
- Any transfer and setup times are assumed to be included in the operation processing time.

4.3 Determination of Job attributes and Scheduling Methods

It was assumed that job information such as slack times, due dates and processing times is very important in making a good schedule. Many studies [15, 16] support this assumption implicitly and as a result, several job characteristics were defined : total available time over total processing time (TA/TP), variance of due dates, variance of slack times, variance of processing times, and so on.

Due to the limitation in current scheduling systems, heuristic methods were used rather than analytic methods. In particular, five loading rules out of many possible heuristic rules were selected based on the fact that they are closely related to important job characteristics, such as processing times and due dates. The scheduling strategies used by SBMRP, forward, backward, were then considered. Combined loading methods (CLMs) were then defined to obtain better performance. These methods were derived by combining the scheduling strategies with the five heuristic rules, resulting in 10 CLMs to be studied : forward First-In/First-Out (FIFO), backwardFIFO, forward Earliest Due Date (EDD), backwardEDD, forward Longest Processing Time (LPT), backward LPT, forward Shortest Processing Time (SPT), backward SPT, forward Least Slack (LS) and backward LS.

4.4 KACE Architecture

Since there was no way to acquire knowledge for the KBES, the KACE process was devised as mentioned above. The key idea of the KACE is to develop heuristic rules experimentally in order to generate better schedules instead of relying solely on either forward or backward scheduling. The process of generating the schedules and acquiring knowledge involves the steps shown in <Figure 1>.

4.4.1 Job Generator

Input : Each job has the parameters such as number of parts (six, fixed), processing time distribution (Exponential), due date distribution (Uniform). In addition to the job parameters, the number of jobs should be specified.

Function : Each job has the eight job characteristics such as total available time/total processing time (TA/TP), variance of due dates (varOfDues), variance of slack times (varOfSlacks), variance of processing time (varOfProcTimes), percent of due tightness (tightness), variance of shop load (varOfLoad), and variance of number of operations for machines and jobs. Features such as the use of probability distributions and the random generation of operation sequences are included since such a large number of jobs with varying characteristics are generated.

output : When the job generator is given the number of jobs, it generates a data file, Jobs, with one record for each job. Each job in the data base is described by its number of parts, processing sequence, and the processing time for each part, and the due date for each part.

4.4.2 Job Executer

Input : The database Jobs, described in the previous section, was given to the job executer.

Function : The job executer applied 10 CLMs to each of the jobs in the jobs database.

output : The job generator produced 10 schedules for each job, based on the 10 CLMs. Since storing these schedules would have required substantial amounts of computer memory, they were sent to the job evaluator immediately.

4.4.3 Job Evaluator

Input : The job evaluator was given the 10 schedules generated from the job executer for each job. After evaluating these schedules, the evaluator received the 10 schedules generated from the next job.

Function : Once schedules had been generated for each job, the schedules were evaluated on the basis of weighted sums of the late and holding costs. The 10 CLMs were ranked in terms of total cost for each job and only the best were saved for further analysis.

output : The evaluator is responsible for generating a Job Summary database which contained the job characteristics and the best CLM for each of the jobs generated.

4.4.4 Rule Generator(C4.5)

Input : Two input files were required, including a file specifying the names and characteristics of the job attributes and the Job Summary database. The former included the definition of classes and attributes for Job Summary. Job Summary was also given to the C4.5 rule generator.

Function : First, the C4.5 decision tree generator was used to produce a decision tree based on the two input files. The decision tree was then input into the C4.5 production rule generator, resulting in a set of production rules.

output : A set of production rules along with its classification performance was obtained.

Each job was assumed to have the job characteristics defined above. Each job was generated based on several different sets of job parameters and its job characteristics were then calculated. After generating a large set of jobs, 10 CLMs were applied to each of them. This resulted in 10 schedules which were evaluated based on criteria minimizing the sum of late and holding costs. An induction program, C4.5 which is a extended version of ID3, was given the results (examples) : the job characteristics and the best CLM(s) for each job. C4.5 induced classification rules in the form of decision tree from the given set of examples and then generated a set of production rules from the decision trees. Based on the test repeated by the number of training instances, the rule analyzer chose 30 rules acquired from the 480 examples as the final set of production rules. The derived knowledge, in rule form, was incorporated into the KBES.

4.4.5 Results

The KBES which consists of 30 production rules was implemented in the Smalltalk system and two rules of them are shown in <Table 1>. The KBES was compared with the two existing scheduling methods of SBMRP(forward and backward) using total cost as a criterion. The job generator generated a large set of test jobs. The three scheduling methods, the KBES and the two existing methods, were applied to this set of test jobs. The KBES first evaluated each job and recommended a CLM based on the job's characteristic. The job executer used this CLM and the standard forward and backward FIFO CLMs to generate three schedules for the job. The job evaluator then evaluated theses schedules in terms of total cost.

<Table 1> Example of selected rules in the Smalltalk system

SBMRPExpert add : (Rule number : 1 condition : [(# taOVERtp lt : 1.17) & (# varOfDues gt : 30.14) & (# tightness lt : 50)] action : [# forwardSPT])	SBMRPExpert add : (Rule number : 2 condition : [(# taOVERtp gt : 1.46) & (# taOVERtp lt : 1.66) & (# varOfSlacks gt : 12.54) & (# varOfSlacks lt: : 6.8) & (# varOfProcTimes gt : 7.94) & (# varOfProcTimes lt : 27.1) & (# varOfLoad gt : 35.13)] action : [# backwardSPT])

The three scheduling methods were evaluated in this manner on 480 unseen test jobs for each of 10 iterations. Experimental results are summarized in <Table 2>. As an average over the 10 iterations, the KBES produced the best schedules for 60% of the jobs, while backward scheduling and forward scheduling methods produced the best schedules for 25% and 15% of the jobs, respectively.

<Table 2> Summary of basic statistics, existing methods and KBES

	Forward Scheduling	Backward Scheduling	KBES
Average (no. of best ranked)	76	126	306
Variance (no. of best ranked)	6.7	46.9	44.6
Percent (no. of best ranked)	15.0	24.8	60.2
Average (total cost)	21,071	17,999	13,612
Standard deviation (total cost)	335	567	398

5. Conclusion

As mentioned, expert system approach can fail because an expert is not available or because there is no expert. As a way to solve this problem, an operator accumulates experience through a simulated system, and then builds an expert system using this newly acquired expertise. However, problem characteristics such as problem attributes and the number of attributes may limit the potential effectiveness of that approach. Furthermore, it is almost impossible to enumerate all possible cases because of system and job dependent and to decide what kind of attributes we use for NP-complete hard problems.

Since there was no way to build an KBES under this situation, a method for KACE was devised. The architecture of KACE consists of five components : a job generator, an executer, an evaluator, a rule generator and an expert system. For the purpose of demonstrating how the architecture of KACE can be implemented, a combinatorial problem was chosen and exemplified for building the KBES.

The combinatorial problem, belonging to the class of NP-complete or NP-hard problems, is a scheduling problem for manufacturing systems, and this results in significant limitations in the solution of real problems. The KBES constructed through the architecture of KACE was compared with the two existing scheduling methods. Experimental results show that the KBES outperforms the two existing methods. This study shows how successfully KACE architecture can be applied to build the KBES in case existing expert system approach fails like the NP-complete hard problem.

References

[1] Akgobek, O., Aydin, Y. S., Oztemel, E., and Aksoy, M. S.; "A New Algorithm for Automatic Knowledge Acquisition in Inductive Learning," *Knowledge-Based System*, 19(6), 2006.

[2] Bell, M. Z.; "Why Expert Systems Fail," *J. Opl. Res. Soc.*, 36(7), 1985.

[3] Berry, D. C.; "The Problem of Implicit Knowledge," *Expert Systems*, 4(3), 1987.

[4] Cohen, P. R. and Feigenbaum, E. A.; The Handbook of Artificial Intelligence, William Kaufmann, 3, 1982.

[5] Hart, A., "The Role of Induction in Knowledge Elucidation," *Expert Systems Journal*, 2(1), 1985.

[6] Hastings, N. A. J., marshall, P., and Willis, R. J.; "Schedule-Based MRP : An Integrated Approach to Production Scheduling and MRP," *J. Opl. Res., Soc.*, 33, 1982.

[7] Hunt, E. B., Martin, J., and Stone, P. J.; Experiments in Induction, New York: Academic Press, 1966.

[8] Kempf, K. G.; "Manufacturing Planning and Scheduling : Where We Are nand Where We Need to Be," 5th Conf. on AI, 1989.

[9] Kim, J., "An Expert System for FMS Schedulimg : Knowledge Acquisition and Development," Ph.D. Thesis, Oregon State Univ., 1988.

[10] Kumara, S. R. T., Joshi, S., Kashyap, Moodie, C. L., and Chang, T. C., "Expert Systems in Industrial Engineering," *Int. J. Prod. Res.*, 24(5), 1986.

[11] Mitchell, T. M., Version Spaces : An Approach to Concept Learning, Ph.D. Dissertation, Rep. No., STAN-CS-78-711, Depart. of CS, Stanford Univ., 1978.

[12] Olson J. R. and Rueter, H. H.; "Extracting Expertise from Experts : Methods for Knowledge Acquisition," *Expert Systems*, 4(3), 1987.

[13] Pigford, D. V, and Baur, G. R.; Expert Systems for Business, Boyd and Fraser Pub Co., 1995.

[14] Quinlan, J. R., "Learning Efficient Classification Procedures and Their Application to Chess End Games," In Michalski, Carbonell and Mitchell(eds.), Machine Learning : An Artificial Intelligence Approach, Palo Alto : Tioga Pub. Company, 1983.

[15] Rinnooykan, A. H. G.; Scheduling Problems : Classification, Complexity and Computations, Nijhoff, The Hague, Holland, 1976.

[16] Rochette, R. and Sadowski, R. P.; "A Statistical Comparison of the Performance of Simple Dispatching Rules for a Particular Set of Job Shops," *Int. J. Prod. Res.*, 14, 1976.

[17] White, C.; Modelling and Design of FMSs, ed. by Kusiak, Elsevier Science, Amsterdam, The Netherlands, 1986.