"This is the peer reviewed version of the following article: Lu J, Zheng Z, Zhang G, He Q, Shi Z (*2015*), A new solution algorithm for solving rule-sets based bilevel decision problems. Concurrency Computation Practice and Experience volume 27 issue 4 pages 830-854 doi: 10.1002/cpe.2833 which has been published in
final form at 10.1002/cpe.2833. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Self-Archiving."

A New Solution Algorithm for Solving Rule-Sets based Bi-level Decision Problems

Jie Lu[1], Zheng Zheng[2], Guangquan Zhang[1], Qing He[3], Zhongzhi Shi[3]
[1]Centre for Quantum Computation and Intelligent Systems
School of Software, Faculty of Information Technology, University of Technology, Sydney,
PO Box 123, Broadway, NSW 2007 Australia
[2]Department of Automation, Beijing University of Aeronautics and Astronautics, Beijing, China
[3]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

Abstract: Bi-level decision addresses compromises between two interacting decision entities within a given hierarchical complex systems. Bi-level programming typically solves bi-level decision problems. However, formulation of objectives and constraints in mathematical functions is required, which are difficult, and sometimes impossible, in real-world situations due to various uncertainties. Our study develops a rule-set based bi-level decision approach, which models a bi-level decision problem by creating, transforming and reducing related rule sets. This study develops a new rule-sets based solution algorithm to obtain an optimal solution from the bi-level decision problem described by rule sets. A case study and a set of experiments illustrate both functions and the effectiveness of the developed algorithm in solving a bi-level decision problem.

Key Words: Rule sets, bi-level decision making, decision support systems, decision modeling, rough sets.

I. INTRODUCTION

Bi-level decision techniques aim to partition control over the decision variables of an optimization problem in a complex systems between the two levels of decision entities. The decision entity at the upper level, known as the leader, will influence or induce the behaviour of the decision entity at the lower level, called the follower, but not completely control its actions. In addition, the follower gains its objective within a given region. In a bi-level decision situation, the decision entity at each level has individual payoff functions, but the decision of the upper level is global. Therefore, the final solution of a bi-level decision problem reflects the leader's goal and also considers the reaction of the follower.

The field of bi-level decision problems has been extensively investigated and highly developed in both theories and applications since first introduced by Von Stackelberg [1], such as [2-7]. A number of bi-level programming models (both linear and nonlinear) and solution methods have been proposed, for example: the *K*th-Best approach [8; 9] and the Kuhn-Tucker approach [4; 9; 10]; intelligent approaches for solving linear bi-level programming problems [11; 12]; the penalty function approach [13; 14]; the stability based approach [15]; and a globally convergent approach for solving non-linear bi-level programming problems [16]. Mathematicians, economists, engineers and other researchers and developers have delivered contributions to this field [17- 20].

However, we have found that this interest often stems from the inherent complexity and consequent uncertainty that exists in a bi-level decision problem. In general, there are two main types of uncertainties in modeling a bi-level decision problem. One is that the parameter values in the objective and constraint functions of the leader and the follower may be uncertain or inaccurate. Some research, such as Sakawa and Nishizaki [21], Sakawa and Yauchi [22], and Zhang and Lu [23] have developed several types of fuzzy optimization approaches to handle this issue. Another type of uncertainty involves the establishment of the objective and constraint functions. That is, how do we determine the relationships between proposed decision variables and formulate these functions for a real bi-level decision problem. The literature, and our experiences, show that some real world bi-level decision problems are difficult, or even impossible, to

be described by mathematical programming models [24, 25]. The challenge remains to be resolved by exploring new methods.

This study considers the challenge of developing a rule-sets based bi-level decision approach to model and solve a bi-level decision problem when it cannot be modeled and solved by mathematical programming. We first proposed an initial rule-sets based modeling algorithm [26] and then a rule-sets based solution algorithm (transformation-based solution algorithm) [27]. This paper proposes a new solution algorithm to solve rule-sets based bi-level decision problems and this algorithm has more advantages in complexity and testing results.

Following the Introduction, Section II introduces the concepts and notions of information tables, rule sets, rule trees and rule comparison. Based on these basic concepts and notions, Section III presents a rule-sets based bi-level decision modeling algorithm, including how to formulate objectives and constraints of a bi-level decision problem by rule sets, and how to apply rule generation and rule reduction algorithms, respectively, to finalize a rule-sets based bi-level decision model. In Section IV, a rule-sets based solution algorithm is given to solve a bi-level decision problem that is described by a set of rule sets. A case based example illustrates the effectiveness of the proposed rule-sets based bi-level decision approach. Experimental results about the effectiveness of the modeling algorithm and the solution algorithms are presented in Section V. Finally, conclusions and future work are outlined in Section VI.

II. BASIC NOTIONS

For the description of the proposed rule-sets based bi-level decision approach, we first introduce some basic notions of information tables, formulas, rules, decision rule set functions and rule trees. We then provide some related definitions and theorems, which will be used in the following sections.

A. Information tables and decision tables

In general, an *information table* is a knowledge expressing system which can be used to represent and process knowledge in machine learning, data mining and other related fields. It provides a convenient way to describe a finite set of objects called the universe by a finite set of attributes [28].

Definition 2.1 [28] (Information table): An information table can be formulated as a tuple:

$$S = (U, At, L, \{V_a \mid a \in At\}, \{I_a \mid a \in At\}),$$

where U is a finite non-empty set of objects, At is a finite non-empty set of attributes, L is a language defined using attributes in At, V_a is a non-empty set of values for $a \in At$, I_a: $U \rightarrow V_a$ is an information function. Each information function I_a is a total function that maps an object of U to exactly one value in V_a.

A *decision table* is a special case of an information table. It is commonly viewed as a functional description, which maps inputs (conditions) to outputs (actions) without necessarily specifying the manner in which the mapping is to be implemented [29; 30].

Definition 2.2 [28] **(Decision table):** A decision table is an information table for which the attributes in A are further classified into disjoint sets of condition attributes C and decision attributes D, i.e. $At = C \cup D$, $C \cap D = \Phi$.

A decision table can be seen as a special and important knowledge expression system. It shows that, when some conditions are satisfied, decisions, actions, operations, or controls can be made. Decision attributes in a decision table may, or may not be unique. For the latter, the decision table can be converted to one with unique decision attributes [31]. Therefore, in this paper, we assume that there is only one decision attribute in a decision table.

B. Formulas and rules

Usually, the knowledge implicated in information tables is expressed by rules. As formulas are the components of rules we first introduce the definition of formulas.

Definition 2.3 [32] **(Formulas):** In the language L of an information table, an atomic formula is given by (a, v), where $a \in At$ and $v \in V_a$. If ϕ and φ are formulas, then so as $\neg\phi$, $\phi \wedge \varphi$, and $\phi \vee \varphi$.

Here, "(a, v)" is a term where a is an attribute and v one of its values. The term covers objects of the information table when the attribute a in At has value v. The semantics of the language L can be defined in the Tarski's style [33] through the notions of a model and satisfactoriness. The model is an information table S, which provides interpretation for symbols and formulas of L.

Definition 2.4 [32] **(Satisfactoriness of formulas):** The satisfactoriness of a formula ϕ by an object x, written $x \models_S \phi$ or in short $x \models \phi$ if S is defined by the following conditions:

(1) $x \models (a, v)$ iff $I_a(x)=v$,

(2) $x \models \neg\phi$ iff not $x \models \phi$,

(3) $x \models \phi \wedge \varphi$ iff $x \models \phi$ and $x \models \varphi$,

(4) $x \models \phi \vee \varphi$ iff $x \models \phi$ or $x \models \varphi$.

If ϕ is a formula, the set

$$m_S(\phi)=\{x \in U \mid x \models \phi\}$$

is called the meaning of the formula ϕ in S. If S is understood, we simply write $m(\phi)$.

The meaning of a formula ϕ is therefore the set of all objects having the property expressed by the formula ϕ. In other words, ϕ can be viewed as the description of the set of objects $m(\phi)$. A connection between the formulas of L and sub-sets of U is established.

To illustrate this idea, we consider an information table given by Table 2.1 [34]. The following expressions are some of the formulas of the language L:

(**height**, tall), (**hair**, dark),

(**height**, tall)$\wedge$(**hair**, dark),

(**height**, tall)$\vee$(**hair**, dark).

The meanings of the formulas are given by:

m((**height**, tall))=$\{o_3, o_4, o_5, o_6, o_7\}$,

m((**hair**, dark))=$\{o_4, o_5, o_7\}$,

m((**height**, tall) $\wedge$(**hair**, dark))=$\{o_4, o_5, o_7\}$,

m((**height**, tall) $\vee$(**hair**, dark))=$\{o_3, o_4, o_5, o_6, o_7\}$.

A *rule* is a statement of the form: "if an object satisfies a formula, then the object must satisfy another formula". The expression of rules can be formulated as follows [28; 32]. A highly related work is due to Apolloni, et al. [35].

Definition 2.5 (Rules): Let $S=(U, At, L, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$ be an information table, then a rule r is a formula with the form

$$\phi \Rightarrow \varphi,$$

where ϕ and φ are formulas of information table S for any $x \in U$,

$$x \models \phi \Rightarrow \varphi \text{ iff } x \models \neg\phi \vee \varphi.$$

Definition 2.6 (Decision Rules): Let $S=(U, C \cup D, L, \{V_a \mid a \in At\}, \{I_a \mid a \in C \cup D\})$ be a decision table, where C is the set of condition attributes and D is the set of decision attributes. A decision rule dr is a rule with the form $\phi \Rightarrow \varphi$, where ϕ, φ are both a conjunction of atomic formulas, for any atomic formula (c, v) in ϕ, $c \in C$, and for any atomic formula (d, v) in φ, $d \in D$.

It is obvious that each object in a decision table can be expressed by a decision rule. The relationship between objects and rules can be defined by the following definition.

Definition 2.7 (Object which is consistent with, or conflicts with, a rule): An object x is said to be consistent with a decision rule dr: $\phi \Rightarrow \varphi$, iff $x \models \phi$ and $x \models \varphi$; x is said to conflict with dr, if $x \models \phi$ and $x \models \neg\varphi$.

C. Decision rule set functions

We introduced the concept of decision rules in Section II (B). We will now explore how to make decisions based on decision rules. We first describe *decision rule set,s* and then define *decision rule set functions*.

Given a decision table $S=(U, At, L, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, where $At = C \cup D$ and $D=\{d\}$, suppose x and y are two variables, where $x \in X$ and $X=V_{a1} \times \ldots \times V_{am}$, $y \in Y$ and $Y=V_d$. V_{ai} is the set of attribute a_i's values and $a_i \in C$, i=1 to m, and m is the number of condition attributes. *RS* is a decision rule set generated from *S*.

Definition 2.8 (Decision rule set function): A decision rule set function *rs* from X to Y is a sub-set of the Cartesian product $X \times Y$, so that, for each x in X, there is a unique y in Y generated with *RS*, so that the ordered pair (x, y) is in *rs*. Here, *RS* is called a decision rule set, x is called a condition variable, y is called a decision variable, X is the definitional domain, and Y is the value domain.

Calculating the value of a decision rule set function is to make decisions for objects with their decision rule sets. In order to present the method for calculating the value of a decision rule set function, we introduce the definition for matching objects to decision rules below.

Definition 2.9 (Matching an object to a decision rule): An object o is said to match a decision rule $\phi \Rightarrow \varphi$, if $o \models \phi$.

Given a decision rule set *RS*, all decision rules in *RS* that are matched by object o are denoted as MR_{RS}^{o}.

This definition provides a brief method for calculating the result of a decision rule set function as follows:

Step 1: Calculate MR_{RS}^{o};

Step 2: Select a decision rule dr from MR_{RS}^{o}, where

$$dr : \wedge\{(a, v_a)\} \Rightarrow (d, v_d);$$

Step 3: Set a decision value of o to be v_d, i.e. $rs(o)=v_d$.

Note that the selection of a decision rule from MR_{RS}^{o} in Step 2 is the key to this process and will impact on the results of Step 3. For example, there is a decision rule set *RS*:

1) $(a, 1) \wedge (b, 2) \Rightarrow (d, 2)$,
2) $(a, 2) \wedge (b, 3) \Rightarrow (d, 1)$,
3) $(b, 4) \Rightarrow (d, 2)$,
4) $(b, 3) \wedge (c, 2) \Rightarrow (d, 3)$,

and an undecided object:

$$o = (a, 2) \wedge (b, 3) \wedge (c, 2).$$

With Step 1, $MR_{RS}^{o} = \{(a, 2) \wedge (b, 3) \Rightarrow (d, 1),$

$$(b, 3) \wedge (c, 2) \Rightarrow (d, 3)\};$$

With Step 2, if we select the final rule as

$$(a, 2) \wedge (b, 3) \Rightarrow (d, 1),$$

Then, at Step 3, $rs(o)=1$;

if we select the final rule as

$$(b, 3) \wedge (c, 2) \Rightarrow (d, 3),$$

Then, at Step 3, $rs(o)=3$.

This example shows that there may be more than one rule in MR^{o}_{RS}. In such cases, when there are multiple decision values of these rules, the result could be controlled according to the above selection method, which is the uncertainty of a decision rule set function. The method for selecting the final rule from MR^{o}_{RS} is therefore very important, and is called the uncertainty solution method. In this paper, we use the AID-based rule tree (Def. 2.11) to deal with this issue, and more details about the method will be discussed in Section II (D) and II (E) below.

D. Rule trees

A rule tree is a compact and efficient structure for expressing a rule set. We use it in this paper as the expression form of rule sets for a bi-level decision model.

Definition 2.10 [36] **(Rule tree):**

(1) A rule tree is composed of one root node, some leaf nodes and some middle nodes;
(2) The root node represents the whole rule set;
(3) Each path from the root node to a leaf node represents a rule;
(4) Each middle node represents an attribute test. Each possible value of an attribute in a rule set is represented by a branch. Each branch generates a new child node. If an attribute is reduced in some rules, then a special branch is needed to represent it and the value of the attribute in this rule is considered as "*", which is different to any possible values of the attribute.

When two nodes are connected with a branch, we call the upper node a start node, and the other, an end node.

Fig. 2.1 gives an example of a rule tree, where "Age", "Educational level (Edulevel)", "Seniority", and "Health" are its conditional attributes, and "Grade" is its decision attribute. The values of these attributes are noted beside branches.

We define the number of nodes between a branch and the root node as the level of the branch (including the root node) in the path. For each rule tree, there are two assumptions:

Assumption 2.1: The branches at the same level represent the possible values of the same attribute. Here, an attribute is expressed by the level of a rule tree.

Assumption 2.2: If a rule tree expresses a decision rule set, the branches at the bottom level represent the possible values of the decision attribute.

Based on Def. 2.10 and the two assumptions, we can improve the rule tree structure by considering the two constraints described in Def. 2.11.

Definition 2.11 (Attribute importance degree (AID) based rule tree): An AID-based rule tree is a rule tree that satisfies the following two additional conditions:

(1) The conditional attribute expressed at the upper level is more important than that expressed at any lower level;

(2) Among the branches with the same start node, the value represented by the left branch is more important (or better) than that represented by any right branch. Each possible value is more important (or better) than the value "*".

In the rule tree illustrated in Fig. 2.1, if we suppose

- ID(a) is the importance degree of attribute a, and
  ID(Age)>ID(Edulevel)>ID(Seniority)>ID(Health);
- (Age, Young) is better than (Age, Middle), and (Age, Middle) is better than (Age, Old);
- (Seniority, Long) is better than (Seniority, Short), and (Health, Good) is better than (Health, Poor),

then the rule tree illustrated by Fig. 2.1 is an AID-based rule tree.

E. Rules comparison

From Section II (C), we know there are some uncertainties when making a decision with decision rule sets. The uncertainty can be eliminated by a process of rule selection. We select a rule correctly only when related information is known. In other words, we are said to be informed only when we can select rules correctly and definitely. This paper presents a rule-tree based model to deal with these kinds of uncertainties. After the ordering of importance degrees and attributes' possible values, a rule tree (Def. 2.10) is improved to become an AID-based rule tree (Def. 2.11). It can be proved that the following theorem holds from the definitions of rules (Def. 2.5) and AID-based rule trees (Def.2.11).

Theorem 2.1: If we suppose the isomorphic trees to be the same, then there is a one-to-one correspondence between a rule set and an AID-based rule tree.

One-to-one correspondence means, each rule set can only construct one AID-based rule tree, and each AID-based rule tree can extract rules to generate one rule set only.

Compared to decision rule sets, an AID-based rule tree has the following advantages:

1) An AID-based rule tree is a more concise data structure, especially when the scale of a rule set is huge;
2) An AID-based rule tree is a more structured data model, and provides useful properties, such as confliction and repetition in an original decision rule set;
3) Using AID-based rule trees can speed up the searching and matching process for decision rules;
4) The rules in an AID-based rule tree are ordered, which provides a way to solve the uncertainty problems in decision rule set functions.

Definition 2.12 (Comparison of rules): Rule $dr_1: \wedge\{(a_i, v_{a1i})\} \Rightarrow (d_1, v_{d1})$ is more important (or better) than rule $dr_2: \wedge\{(a_i, v_{a2i})\} \Rightarrow (d_2, v_{d2})$, if v_{a1k} is more important (or better) than v_{a2k} or the value of a_k is deleted from rule dr_2, where attribute a_i is more important (or better) than a_{i+1}, and for each $j<k$, $v_{a1j}=v_{a2j}$. If each attribute a_i, v_{a1i} has the same importance (or evaluation) degree as v_{a2i}, rule dr_1 has the same importance (or evaluation) degree as rule dr_2.

For example, we have two rules as follows:

dr_1: (Age, Middle) $\wedge$ (Working Seniority, Long) $\Rightarrow$ 2,

dr_2: (Age, Middle) $\wedge$ (Working Seniority, Short) $\Rightarrow$ 3,

and the value "Long" is better than the value "Short" in the attribute "Working Seniority", with Def. 2.12 we know dr_1 is better than dr_2.

Theorem 2.2: In an AID-based rule tree, the rule expressed by the left branch is more important (or better) than the rule expressed by the right branch.

It can be proven from Def. 2.11 and Def. 2.12.

Theorem 2.3: After transformation to an AID-based rule tree, the rules in a rule set are totally in order, that is, every two rules can be compared.

The theorem holds from Def. 2.11 and Theorem 2.2.

For example, we can order the rules expressed by the rule tree shown in Fig. 2.1 as follows:

1) (Age, Young) $\wedge$ (Edulevel, High) $\Rightarrow$ 2,
2) (Age, Middle) $\wedge$ (Working Seniority, Long) $\Rightarrow$ 2,
3) (Age, Middle) $\wedge$ (Working Seniority, Short) $\Rightarrow$ 3,
4) (Age, Old) $\Rightarrow$ 4,
5) (Edulevel, Short) $\Rightarrow$ 4,

where rule i is better than rule i+1, i = 1, 2, 3, 4.

III. A RULE-SETS BASED MODELING ALGORITHM FOR BI-LEVEL DECISION

In general, a bi-level decision problem is described by bi-level programming, in which the objectives and constraints of the leader and the follower are expressed by linear, or non-linear functions as follows:

$$\begin{aligned} &\min_{x \in X} F(x, y) \\ &\text{subject to } G(x, y) \le 0 \\ &\qquad \min_{y \in Y} f(x, y) \\ &\qquad \text{subject to } g(x, y) \le 0 \end{aligned} \tag{3.1}$$

where $x, \in R^n$, $y, \in R^m$.

However, some real-world bi-level decision problems cannot be easily formulated or approximated as a linear or non-linear bi-level programming model. To handle the issue, this study uses rules sets (Def. 2.5), rather than mathematical functions, to express objectives and constraints of a bi-level decision problem. The concepts and theorems presented in Section II show that rule sets can be used to establish suitable bi-level decision models.

A. Objectives and objective decision rule sets

In a bi-level decision problem, both the leader and the follower have individual objectives. When these objectives are hard to describe by functions in mathematical programming models, we will consider expressing them by decision tables (Def. 2.1). In principle, a decision table can implement any computable functions. It is observed that any Turing Machine program can be "emulated" by a decision table by letting each Turing Machine instruction of the form (input, state) + (output, tape movement, state) be represented by a decision rule (or a decision table), where (input, state) are conditions and (output, tape movement, state) are actions. From a more practical point of view, it can also be shown that all computer program flowcharts can be emulated by decision tables [29; 30]. Therefore, after emulating all possible situations in a bi-level decision problem, all objective functions can be transformed to a set of decision tables, named as objective decision tables. That is, the objectives of the leader and the follower in a bi-level decision problem can be transformed into a set of decision tables, where decision variables are represented by the objects in these decision tables.

However, there are two disadvantages in the application of decision tables. One is that when a problem is very complex, its decision tables may become extremely large and related algorithm effectiveness becomes very low. Another is that the objects in a decision table lack adaptability. They are hard to adapt to any new situations and one object can only record a situation. The literature shows that decision rule sets are generally knowledge generated from decision tables and have stronger knowledge expressing ability than decision tables by overcoming the two disadvantages indicated. In the proposed bi-level decision model, we use decision rule (Def. 2.6) sets to represent the objectives of the leader and the follower, called objective decision rule sets, while decision tables can be viewed as special cases of decision rule sets.

B. Constraints and constraint rule sets

In a bi-level decision model, the constraints of each decision entity can be seen as the description of the searching space of the entity in the decision problem, and can be represented by a set of rule sets. Similarly, as discussed above for objectives of a bi-level decision model, after emulating all possible situations in the constraint field, the constraints can be formulated to an information table. When the information table is too big to be processed, it can be transformed to rule sets using the mining association rules methods provided by Agrawal & Srikant [37] Agrawal et al. [38].

Rule sets can be viewed as knowledge generated from information tables, but with stronger knowledge expressing ability and better adaptability than information tables. An information table can be viewed as a special case of rule sets. By using rule sets, we provide the following definition for constraint functions.

Definition 3.1 (Constraint Function): Suppose x is a decision variable and RS is a rule set, thus a constraint function $cf(x, RS)$ is defined as

$$cf(x, RS) = \begin{cases} True, & if \quad \forall r \in RS \quad x \in m(r) \\ False, & else \end{cases}$$

The meaning of the constraint function $cf(x, RS)$ is whether variable x belongs to the region constrained by RS.

C. A rule-sets based bi-level decision model

As described in Equation (3.1), in a bi-level decision model, control of the decision variables is partitioned between the decision entities who seek to optimize their individual payoff functions. Both the leader and the follower are assumed to know the objective and feasible choices available to the other. Therefore, based on the discussions above, we can describe the objectives and constraints of the two decision entities by rule sets, called a rule-sets based bi-level decision model, as follows.

Definition 3.2 (Rule-sets based bi-level decision model):

$$\begin{aligned} &\min_x f_L(x, y) \\ &\text{subject to } cf(x, G_L) = \text{True} \\ &\qquad \min_y f_F(x, y) \\ &\qquad \text{subject to } cf(x, G_F) = \text{True}, \end{aligned} \tag{3.2}$$

where x and y are decision variables (vectors) of the leader and the follower respectively; f_L and f_F are the objective decision rule set functions (Def. 2.8) of the leader and the follower respectively; cf is the constraint function (Def. 3.1); F_L and G_L are the objective decision rule set and constraint rule set of the leader; and F_F and G_F are the objective decision rule set and constraint rule set of the follower, respectively.

We assume the decision rule sets of the objectives will cover the objects in the constraints. That is, each object in the constraints can be matched by one decision rule at least in the objective decision rule sets. The assumption is reasonable, because when the objective decision tables used to generate objective decision rule sets are huge and the objects in these decision tables are uniformly distributed, the resulting decision rule sets usually cover most of the objects to be decided. In other situations, where some objects in the constraint fields cannot be matched by any decision rules in the objective decision rule sets, some additional methods should be introduced, such as similarity matching and fuzzy matching [39]. Therefore, it is acceptable to establish the models and develop related decision methods based on the above assumption in this study.

D. An algorithm for modeling bi-level decision problems by rule sets

Based on the objective decision rule sets, constraint rule sets, and related definitions introduced above and in our previous work [26], we provide a rule-sets based bi-level decision modeling algorithm as follows.

Algorithm 3.1 (Rule-sets based bi-level decision modeling algorithm)

Input: A bi-level decision problem with the objectives and constraints of the leader and the follower;

Output: A rule-sets based bi-level decision model;

Step 1: Transform the bi-level decision problem with a set of rule sets (information tables are as special cases);

Step 2: Pre-process F_L, so as to delete reduplicate rules from the rule sets, eliminate noisy and etc.;

Step 3: If F_L needs to be reduced, then use a reduction algorithm to reduce F_L;

Step 4: Pre-process G_L, so as to delete reduplicate rules from the rule sets, eliminate noise, etc.;
Step 5: If G_L needs to be reduced, then use a reduction algorithm to reduce G_L;
Step 6: Pre-process F_F, so as to delete reduplicate rules from the rule sets, eliminate noise etc.;
Step 7: If F_F needs to be reduced, then use a reduction algorithm to reduce F_F;
Step 8: Pre-process G_F, so as to delete reduplicate rules from the rule sets, eliminate noise etc.;
Step 9: If G_F needs to be reduced, then use a reduction algorithm to reduce G_F.

Complete

The flow chart of the proposed bi-level decision problem modeling algorithm is shown in Fig. 3.1. We now provide explanations for the proposed Algorithm 3.1. Step 1 is the key step for the modeling process. Decision makers (or experts) complete this step by transforming a bi-level decision problem into a set of information tables or related rule sets. This transformation is done by laying out all possible situations of the bi-level decision problem.

In Step 2, Step 4, Step 6 and Step 8, four sets of decision rule sets are pre-processed respectively. As data incompleteness, noise, and inconsistency are the common characteristics for a huge real data set, we need to use related techniques to eliminate these problems before using the rule sets to model a bi-level decision problem [40].

In Step 5, Step 7, and Step 9 of Algorithm 3.1, related rule sets are reduced by applying a reduction algorithm because of at least one of the following three reasons:

(1) When modeling a real-world bi-level decision problem, the rule sets in the model are often at a large scale, which is not convenient to process, and cannot be easily interpreted and understood.

(2) The rules in the rule sets lack adaptability. In this situation, the rule sets cannot adapt well to new situations , so it is unable, or has poor ability, to support decision making.

(3) The rule sets in the model are just original data sets; the patterns in these data sets need to be extracted, and the results are more general rules.

To reduce the size of a decision rule set or extract decision rules from a decision table, several rough sets [41] based decision rule extraction and reduction algorithms, named as value reduction algorithms, have been developed [28; 36; 42- 46]. These algorithms have been successfully applied in many fields [47-49]. There are some rough sets based software systems, such as ROSETTA [50], RIDAS [51], RSES [52], which can be used to reduce the size of a decision rule set or extract decision rule sets from a decision table. Therefore, this issue can be handled by applying these methods and systems.

To reduce the size of the constraint rule sets or to generate constraint rules from the information tables, several effective methods are available, such as Apriori algorithm [38], Fast algorithms for mining association rules [37], and the FP tree algorithm [53]. We select one of them to complete this task. We have found that in many cases, when the constraint rules are obvious or already known, the rule generation process can be passed over. After the pre-process and reduction for these rule sets, the decision rule set functions are improved correspondingly.

Now we analyse the complexity of Algorithm 3.1. Obviously, it can be estimated as the complexity of Step 1, Steps 2, 4, 6, 8 and Steps 3, 5, 7, 9 respectively.

Suppose p_{oL} and p_{oF} are the numbers of the rules in the objective decision rule sets of the leader and the follower generated in Step 1, respectively; p_{cL}, and p_{cF} are the numbers of the rules in the constraint rule sets of the leader and the follower generated in Step 1 respectively; and m_L and m_F are the numbers of the condition attributes of the leader and the follower. For Step 1, the complexity is

$$O((m_L + m_F)(p_{oL} + p_{oF} + p_{cL} + p_{cF})).$$

For Steps 2, 4, 6, 8, different pre-process methods can cause different complexities. For the above mentioned pre-process methods, the complexity is between $O((m_L + m_F)p)$ and $O((m_L + m_F)p^2)$, where $p=p_{oL}$ for Step 2, $p= p_{oF}$ for Step 4, $p= p_{cL}$ for Step 6, and $p= p_{cF}$ for Step 8.

For Steps 3, 5, 7, and 9 the time complexity depends on the sizes of the processed rule sets. Using the methods mentioned above, it has complexity

$$O((m_L + m_F) \cdot p \cdot (p-1)),$$

where $p=p_{oL}$ for Step 3, $p=p_{oF}$ for Step 5, $p=p_{cL}$ for Step 7, and $p=p_{cF}$ for Step 9.

Therefore, Algorithm 3.1 has the maximal time complexity

$$O\left((m_L + m_F)\left(p_{oL}^{\ 2} + p_{oF}^{\ 2} + p_{cL}^{\ 2} + p_{cF}^{\ 2}\right)\right).$$

In the following sub-section, we will apply the proposed algorithm for a case based example to illustrate the modeling process of a bi-level decision problem b.

E. A case based example

A company's recruitment management is distributed in two levels: the upper level is the company's executive committee and the lower is the workshop management committee. Recruitment policy requires that the executive committee mainly consider how to meet the overall business objectives with a long-term development plan, and the workshop management committee concentrate on the current daily needs of workers. Obviously, their objectives are different. However, their objectives are transparent to each other, though they may operate in separate ways. A recruitment action will ultimately emerge that is the optimal result for the company as a whole, which also considers current daily needs. This is a typical bi-level decision problem, in which the company executive committee is the leader, and the workshop management committee, the follower.

When deciding whether a person could be recruited for a particular position, the company executive committee mainly considers the following two factors: "age" and "education level (edulevel)" of the applicant; and the workshop management committee mainly considers another two factors: "seniority" and "health". Suppose the condition attributes, in an ascending order according to the importance degrees, are "age", "edulevel", "seniority", and "health".

Obviously, it's hard for the two committees to express the conditions of the workers whom they want to recruit as linear or non-linear mathematical functions. However, the data for all applicants are available in a database. We can therefore build two decision tables as shown in Tables 3.1 and 3.2 using the data. We then generate decision rule sets from the two decision tables to represent the objectives of the two committees. The condition attributes of the two decision tables are the factors; and the decision attributes of the two decision tables are acceptance grades of these applicants. The constraints of the two committees are expressed by simple rule sets (Equations 3.3 and 3.4), which define the constraint regions.

We use Alg. 3.1 to describe the problem in a rule-sets based bi-level decision model.

Alg. 3.1-Step 1: Transform the problem with rule sets (information tables as special cases).

Table 3.1 and Table 3.2 represent the objective decision rule sets of the leader and the follower, respectively. Equations (3.3) and (3.4) give the constraint rule sets of the leader and the follower, respectively.

The constraint rule set of the leader:

$$G_L = \{\text{True} \Rightarrow (\text{Age, Young}) \vee (\text{Age, Middle})\} \qquad (3.3)$$

The constraint rule set of the follower:

$$G_F = \{\text{True} \Rightarrow (\text{Seniority, Long}) \vee (\text{Seniority, Middle})\} \qquad (3.4)$$

Because the scale of the data is very small, the pre-process steps (Steps 2, 4, 6 and 8) can be passed over. The constraint rule sets of the leader and the follower are brief , so the reduction steps of G_L and G_F (Step 5 and Step 9) can be ignored.

Alg. 3.1- Step 3 and Step 7: Reduce the objective decision rule sets of the leader and the follower.

After reducing the decision tables based on the rough set theory given in Section 2 (E), we find reduced objective decision rule sets of the leader and the follower, as shown in Equations (3.5) and (3.6),

respectively. We use the decision-matrices based value reduction algorithm [54] in the RIDAS system [51] to conduct this job.

The reduced objective decision rule set of the leader:

F_L= {(Age, Young) ∧ (Seniority, Middle) ⇒ (Grade, 2)
(Age, Middle) ∧ (Edulevel, High) ⇒ (Grade, 2)
(Edulevel, Short) ∧ (Seniority, Short) ⇒ (Grade, 4)
(Edulevel, Middle) ∧ (Seniority, Short) ⇒ (Grade, 3)
(Edulevel, Middle) ∧ (Seniority, Long) ⇒ (Grade, 2)
(Age, Old) ∧ (Health, Middle) ⇒ (Grade, 3)
(Age, Old) ∧ (Edulevel, Short) ⇒ (Grade, 4)
(Age, Middle) ∧ (Health, Good) ⇒ (Grade, 3)
(Age, Middle) ∧ (Seniority, Long) ∧ (Health, Middle) ⇒ (Grade, 2)
(Age, Old) ∧ (Edulevel, High) ∧ (Health, Good) ⇒ (Grade, 2)
(Edulevel, High) ∧ (Health, Poor) ⇒ (Grade, 3)
(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Long) ⇒ (Grade, 4)
(Age, Young) ∧ (Edulevel, Short) ∧ (Seniority, Long) ⇒ (Grade, 3)} (3.5)

The reduced objective decision rule set of the follower:

F_F= {(Edulevel, High) ∧ (Health, Good) ⇒ (Grade, 2)
(Edulevel, Short) ∧ (Seniority, Short) ⇒ (Grade, 4)
(Age, Old) ∧ (Health, Middle) ⇒ (Grade, 3)
(Age, Young) ∧ (Seniority, Long) ∧ (Health, Middle) ⇒ (Grade, 4)
(Seniority, Middle) ⇒ (Grade, 3)
(Seniority, Long) ∧ (Health, Good) ⇒ (Grade, 2)
(Seniority, Short) ∧ (Health, Poor) ⇒ (Grade, 4) } (3.6)

With above steps, we get the rule-sets based bi-level decision model of the decision problem, as follows:

$$\min_x f_L(x, y)$$
$$\text{subject to } cf(x, G_L) = \text{True}$$
$$\min_y f_F(x, y)$$
$$\text{subject to } cf(x, G_F) = \text{True},$$

where f_L, f_F are the corresponding decision rule set functions based on rule sets F_L, F_F , respectively.

This example shows the feasibility to model a bi-level decision problem by the proposed rule-sets based modeling algorithm.

IV. A NEW RULE-SETS BASED SOLUTION ALGORITHM FOR BI-LEVEL DECISIONS

In this section, we will present a solution algorithm to solve a bi-level decision problem that is described by rule sets.

A. Concepts

Before a rule-sets based solution algorithm is given, we first introduce the concept of a solution for bi-level decision problems and analyze related properties. Based on the solution definition of bi-level optimization given by Bard [6], we have the following solution concepts for a bi-level decision problem described by rule sets:

Definition 4.1 (Bi-level decision solution):

(a) Constraint region of a bi-level decision problem:

$$S= \{(x, y): \ cf(x, G_L) = \text{True}, \ cf(x, G_F) = \text{True}\}$$

(b) Feasible set for the follower for each fixed x:

$$S(x) = \{y: (x, y) \in S\}$$

(c) Projection of S onto the leader's decision space:

$$S(X) = \{x: \exists y, (x, y) \in S\}$$

(d) Follower's rational reaction set for $x \in S(X)$:

$$P(x) = \{y : y \in \arg\min_y [f_F(x, y') : y' \in S(X)]\}$$

(e) Inducible region:

$$IR = \{(x, y): (x, y) \in S, y \in P(x)\}$$

To ensure that the model presented by Equation (3.2) is well posed it is common to assume that S is non-empty and compact, and that for all decisions taken by the leader, the follower has some room to respond, i.e. $P(x) \neq \Phi$. The rational reaction set $P(x)$ defines the response, while the inducible region IR represents the set over which the leader may optimize. In terms of the above notation, the bi-level decision problem can be written as

$$\min \{ f_L(x, y) : (x, y) \in IR\}.$$

From the features of bi-level decisions, once the leader selects an x, the first term in the follower's objective function becomes a constant and can be removed from the problem. In this case, we replace $f_F(x, y)$ with $f_F(y)$.

To begin, let $(x_{[1]}, y_{[1]})$, $(x_{[2]}, y_{[2]})$, …, $(x_{[N]}, y_{[N]})$ denote N ordered feasible solutions to a rule-sets based one level one objective problem

$$\min_x \{ f_L(x, y) : (x, y) \in S\} \tag{4.1}$$

so that

$$f_L(x_{[i]}, y_{[i]}) \leq f_L(x_{[i+1]}, y_{[i+1]}) \tag{4.2}$$

and i=1, …, N-1. Solving Equation (4.1) is equivalent to finding the global optimum $(x_{[k]}, y_{[k]})$. This result will be used in the following algorithm.

B. An algorithm for solving rule-sets based bi-level decision problems

We now present an algorithm for solving a rule-sets based bi-level decision problems. The main objective of the algorithm is to repeatedly solve two rule-sets based one-level decision problems. One is for the leader in all of the variables x and a sub-set of the variables y associated with an optimal basis to the follower's problem. The other is for the follower with all of the variables x fixed. The leader first makes his/her decision, and the decision will influence the objective decision rule-set functions and the constraint rule-set functions of the follower. In a systematic way, the algorithm explores the optimal solution of the follower's problem for x fixed, and then returns to the leader's problem with the corresponding variables y. If the set of variables y are not an optimal solution to the leader's decision problem, then the leader modifies its decision and repeats the procedures.

Based on the analysis in Section II (E), we know that there is a one-to-one correspondence between a rule set and an AID-based rule tree (Theorem 2.1). Using AID-based rule trees to represent decision rule sets can solve the uncertainty problems arising in decision making (Section II (C)). Therefore, all the rule sets are expressed by AID-based rule trees in this algorithm.

Algorithm 4.1 (A rule-sets based bi-level decision solution algorithm)

Input: The objective decision rule set $F_L = \{dr_{L1}, \ldots, dr_{Lp}\}$ and the constraint rule set G_L of the leader, the objective decision rule set $F_F = \{dr_{F1}, \ldots, dr_{Fq}\}$ and the constraint rule set G_F of the follower.

Output: The final decision (optimal solution) of the leader and the follower ($x_{[i]}$, $y_{[i]}$).

Step 1: Construct the objective rule tree FT_L of the leader by F_L;

Step 1.1: Arrange the condition attributes in ascending order according to the importance degrees of these attributes. Let these attributes be the discernible attributes of levels from the top to the bottom of the tree;

Step 1.2: Initialize FT_L to an empty AID-based rule tree;

Step 1.3: For each rule R of the objective decision rule set F_L {

Step 1.3.1: let CN←root node of the rule tree FT_L;

Step 1.3.2: For i=1 to m /*m *is the number of levels in the rule tree*/

{If there is a branch of CN representing the ith discernible attribute value of rule R, then

let CN←node I; /*node I is the node generated by the branch*/

else {Create a branch of CN to represent the ith discernible attribute value;

According to the value order of the ith discernible attribute, place the created branch in the right place;

Let CN←node J /*node J is the end node of the branch*/}}}

Step 2: Construct the objective rule tree FT_F of the follower by F_F;

The detail of Step 2 is similar to that in Step 1. That is, it can be described by the same sub-steps of Step 1, but replacing FT_L with $FT_{F,}$ and F_L with F_F .

Step 3: Solve the problem shown in Equation (4.1) to obtain the optimal solution ($x_{[i]}$, $y_{[i]}$) , and initialize i=1;

Step 3.1: Initialize FT_L' to an empty AID-based rule tree, where FT_L' represents the objective rule tree of the follower pruned by constraint rule sets;

Step 3.2: Use the constraint rule tree sets G_L and G_F to prune FT_L;

For each rule dr in G_L and G_F,

Add the rules in FT_L that are consistent with dr to FT_L';

Delete the rules in FT_L and FT_L' that conflict with dr;

Step 3.3: Search for the rules with the minimal decision value in FT_L' and the resulting rule set is RS= {dr_1, …, dr_m}, where dr_1 to dr_m are the rules from left to right in FT_L';

Step 3.4: Let dr be the first rule in RS;

Step 3.5: RS=RS-{dr};

OS= {o|o is the object consistent with dr and the constraint rule sets};

Step 3.6: Order the objects in OS so that the ith object in OS is better than the (i+1)th object in OS according to Def. 2.12;

Step 3.7: The solution to the problem shown in Equation (4.1) is the first object (Def. 2. 12) in OS.

Step 4: Prune the objective rule tree of the follower with the constraint rule sets of the leader and the follower, and suppose FT_F ' is the resulting objective rule tree;

Step 5: Prune the rules from FT_F', which are not consistent with the rule True$\Rightarrow x_{[i]}$and suppose the result is a rule tree FT_F'';

Step 6: Solve the follower's rule-sets based decision problem:

$$\min_{y}\{ f_F(x_{[i]}, y) : y \in P(x_{[i]})\} \tag{4.3}$$

and find the optimal solution ($x_{[i]}$, y);

Step 6.1: Search for the rules with the minimal decision value in FT_F'' and the resulting rule set is RS'= {dr_1', …, dr_m'};

Step 6.2: Let dr' be the first rule in RS';

Step 6.3: RS'=RS'-{dr'}; OS'= {o'|o' is the object consistent with dr' and the constraint rule sets};

Step 6.4: Order the objects in *OS'* so that the *i*th object in *OS'* is better than the (*i*+1)th object in *OS'* according to Def. 2.12;

Step 6.5: The solution of the follower's problem is the first object (Def. 2.12) in *OS'*.

Step 7: If $y= y_{[i]}$, then

{The optimal solution set is obtained, which is $(x_{[i]}, y_{[i]})$; End;}

Else { Go to Step 8 ;}

Step 8: Select another solution for the follower.

Step 8.1: *OS'=OS'*-{$(x_{[i]}, y)$};

Step 8.2: If *OS'* is null, then

{If *RS'* is null, then

{Go to Step 9;}

Else

{Let *dr'* be the first rule in *RS*;

RS'=RS'-{*dr'*};

OS'= {*o'*|*o'* is the object consistent with *dr'* and the constraint rule sets};

Step 8.3: Order the objects in *OS'* and the next solution of the follower's problem is the first object $(x_{[i]}, y)$ (Def. 2.12) in *OS'*.

Step 8.4: Go to Step 7;

Step 9: Select another solution for the leader.

Step 9.1: *OS=OS*-{$(x_{[i]}, y_{[i]})$};

Step 9.2: $W=W\cup\{(x_{[i]}, y_{[i]})\}$;

Step 9.3: If *OS* is null, then

{If *RS* is null, then {There isn't an optimal solution for the problem; End;}

else

{ Let *dr* be the first rule in *RS*;

RS=RS-{*dr*};

OS= {*o*|*o* is the object consistent with *dr* and the constraint rule sets;}

Let $(x_{[i+1]}, y_{[i+1]})$ be the first object in *OS*;

i=i+1;

Go to Step 5;}

Complete

Suppose n_{oL} and n_{oF} are the numbers of the rules in the objective decision rule sets of the leader and the follower, respectively, and n_{cL}, n_{cF} are the numbers of the rules in the constraint rule sets of the leader and the follower, respectively. $n_{oL}^{\min}$ and $n_{oF}^{\min}$ are the numbers of the rules with the minimal decision value in FT_L *' and* FT_F *,'* respectively. Suppose $c_{L1}, \cdots, c_{Lm_L}$ are the leader's condition attributes, where m_L is the number of the leader's condition attributes, and $c_{F1}, \cdots, c_{Fm_F}$ are the follower's condition attributes, where m_F is the number of the follower's condition attributes. For each attribute c, d_c is the number of the possible values of the attribute. Then, the total maximal time complexity of the algorithm is:

$$O\left((n_{oL}+n_{oF})(n_{cL}+n_{cF})(m_L+m_F)+n_{oL}^{\min}\cdot\prod_{i=1}^{m_L}d_{c_{Li}}\cdot\prod_{i=1}^{m_F}d_{c_{Fi}}+n_{oF}^{\min}\cdot\prod_{i=1}^{m_F}d_{c_{Fi}}\right). \tag{4.4}$$

The total minimal time complexity of the algorithm is $O((n_{oL}+n_{oF})(n_{cL}+n_{cF})(m_L+m_F))$. In Eq.(4.4), $\prod_{i=1}^{m_L}d_{c_{Li}}\cdot\prod_{i=1}^{m_F}d_{c_{Fi}}$ in the second item is the maximal number of objects in *OS* (in Step 3.5), and $\prod_{i=1}^{m_F}d_{c_{Fi}}$ in the third item is the maximal number of objects in OS' (in Step 6.3). Usually, the numbers of objects in *OS* and

OS' are much smaller than $\prod_{i=1}^{m_L} d_{c_{Li}} \cdot \prod_{i=1}^{m_F} d_{c_{Fi}}$ and $\prod_{i=1}^{m_F} d_{c_{Fi}}$, respectively. $n_{oL}^{\min}$ and $n_{oF}^{\min}$ are very small in most cases. Table 4.1 is the comparison of time complexities between Algorithm 4.1 and the transformation-based algorithm presented in [27]. The minimal time complexity of Alg.4.1 is much smaller than that of the transformation-based solution algorithm. In a situation where possible values of attributes are few, the maximal time complexity of Alg.4.1 will be lower than that of the transformation-based solution algorithm. The space complexity of Alg.4.1 is $O\left(\left(n_{oL}+n_{oF}+n_{cL}+n_{cF}\right)\left(m_L+m_F\right)\right)$. Fig. 4.1 shows the flow charts of Alg. 4.1. We will use the recruitment case study to further illustrate the use of the algorithm.

C. A case based example

In Section III (E), we established a rule-sets based bi-level decision model for a company recruitment situation. Now, we use Alg. 4.1 to solve the recruitment bi-level decision problem, to find a solution for this rule-sets based bi-level decision model established. We suppose the four condition attributes used in this case are ordered by attribute importance degrees as "age", "edulevel", "seniority", and "health" for the recruitment issue.

Alg. 4.1-Step 1: Construct the objective rule tree FT_L of the leader by F_L and the result is shown in Fig. 4.2;

Alg. 4.1-Step 2: Construct the objective rule tree FT_F of the follower by F_F and the result is shown in Fig. 4.3;

Alg. 4.1-Step 3: Solve problem shown in Equation (4.1), and initialize i=1.

Step 3.1: Let FT_L' be the objective rule tree of the follower pruned by the constraint rule sets, and initialize FT_L' to an empty AID-based rule tree;

Step 3.2: Use the constraint rule tree GT_L to prune FT_L and the result is FT_L' as shown in Fig. 4.4;

Step 3.3: Search for the rules with the minimal decision value in FT_L' and the resulting rule set is

RS={(Age, Young) ∧ (Seniority, Middle)⇒ (Grade, 2);
(Age, Middle) ∧ (Edulevel, High)⇒ (Grade, 2);
(Age, Middle) ∧ (Seniority, Long) ∧ (Health, Middle)⇒(Grade, 2);
(Edulevel, Middle) ∧ (Seniority, Long) ⇒(Grade, 2)};

Step 3.4:

dr: (Age, Young) ∧ (Seniority, Middle)⇒(Grade, 2);

Step 3.5-3.6:

RS={(Age, Middle) ∧ (Edulevel, High)⇒ (Grade, 2);
(Age, Middle) ∧ (Seniority, Long) ∧ (Health, Middle)⇒(Grade, 2);
(Edulevel, Middle) ∧ (Seniority, Long) ⇒(Grade, 2)};

OS={(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Poor);
(Age, Young) ∧ (Edulevel, Middle) ∧ (Seniority, Middle) ∧ (Health, Good);
(Age, Young) ∧ (Edulevel, Middle) ∧ (Seniority, Middle) ∧ (Health, Middle);
(Age, Young) ∧ (Edulevel, Middle) ∧ (Seniority, Middle) ∧ (Health, Poor);
(Age, Young) ∧ (Edulevel, Poor) ∧ (Seniority, Middle) ∧ (Health, Good);
(Age, Young) ∧ (Edulevel, Middle) ∧ (Seniority, Middle) ∧ (Health, Middle);
(Age, Young) ∧ (Edulevel, Middle) ∧ (Seniority, Middle) ∧ (Health, Poor)};

Step 3.7: The solution of problem (4.1) is the first object in *OS*, that is

o = (Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good).

Alg. 4.1-Step 4: Let FT_F' be the objective rule tree of the follower pruned by the constraint rule sets and FT_F' is as shown in Fig. 4.5. $W = \varphi$;

Alg. 4.1-Step 5: $x_{[1]}$ = (Age, Young) ∧ (Edulevel, High). Prune the rules from FT which are not consistent with

$$\text{True} \Rightarrow x_{[1]}$$

and suppose the result is the rule tree FT' as shown in Fig. 4.6;

Alg. 4.1-Step 6: Solve the follower's rule-sets based decision problem below.

$$\min_{y}\{ f_F(x_{[1]}, y) : y \in P(x_{[1]})\} .$$

Step 6.1: Search for the rules with the minimal decision value in $FT''$and the resulting rule set is

RS' = {(Edulevel, High) ∧ (Health, Good) ⇒ (Grade, 2) ;
(Seniority, Long) ∧ (Health, Good) ⇒ (Grade, 2)};

Step 6.2:

dr': (Edulevel, High) ∧ (Health, Good) ⇒ (Grade, 2);

Step 6.3-6.4:

RS'= {(Seniority, Long) ∧ (Health, Good) ⇒ (Grade, 2)};

OS'= {(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Long) ∧ (Health, Good);
(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good);
(Age, Middle) ∧ (Edulevel, High) ∧ (Seniority, Long) ∧ (Health, Good);
(Age, Middle) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good)};

Step 6.5: The solution of the follower's problem is
(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Long) ∧ (Health, Good).

Alg. 4.1-Step 7: Because $y \neq y_{[i]}$, Go to Step 8;

Alg. 4.1-Step 8:

Step 8.1-8.2:

OS'= {(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good);
(Age, Middle) ∧ (Edulevel, High) ∧ (Seniority, Long) ∧ (Health, Good);
(Age, Middle) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good)};

Step 8.3: The next solution of the follower's problem is

o' = (Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good);

Step 8.4: Go to Step 7;

Alg. 4.1-Step 7: Because $y = y_{[i]}$, the optimal solution is
(Age, Young) ∧ (Edulevel, High) ∧ (Seniority, Middle) ∧ (Health, Good).

Complete

The solution shows that the applicants who are young, with a high education level, middle seniority experience, and good health will most closely meet the requirements of the company. This solution meets the objective of the company executive committee by meeting the objectives of the workshop management committee and handling its reaction for each possible recruitment decision of the company executive committee. This case study also illustrates the functions and effectiveness of the proposed rule-sets based solution algorithm to solve bi-level decision problems.

V. EXPERIMENTS

In order to test the effectiveness of the proposed rule-sets based bi-level decision problem modeling algorithm (Algorithm 3.1) and solution algorithm (Alg. 4.1), we implemented these two algorithms within Matlab 6.5. We then used a set of classical data sets in the UCI database

(http://www.ics.uci.edu/~mlearn/MLRepository.html) to conduct two experiments for the purpose. The UCI database consists of many data sets collected from real applications that can be used by the decision systems and machine learning communities for related empirical analysis of algorithm effectiveness tests.

For each data set chosen, we selected the first half as the original objective decision rule set of the leader, and the remaining half for the original objective decision rule set of the follower. We assumed that there were no constraints, which means all objects consistent with the objective decision rule sets were in the constraint region. The importance degrees of the condition attributes were descending from the first condition attribute to the last condition attribute. The two experiments are processed on the computer with 2.33GHz CPU and 2G memory space.

Experiment 1: Testing of Algorithm 3.1 with the data sets in UCI database.

Step 1. Randomly choose 50% of the objects from the data set to be the original objective decision rule set of the leader, and the remaining 50% of the objects to be the original objective decision rule set of the follower;

Step 2. Apply Algorithm 3.1 to construct a rule-sets based bi-level decision model by using the chosen rule sets. Here, we use the decision matrices based value reduction algorithm [54] in the RIDAS system [51] to reduce the sizes of original rule sets.

Experiment 2: Testing of Algorithm 4.1 and the transformation-based solution algorithm [27] with the data sets in UCI database.

Following Steps 1 and 2 in Experiment 1, we have

Step 3. Apply Algorithm 4.1 or transformation-based solution algorithm to find a solution from the generated rule-sets based bi-level decision model in Experiment 1.

The complexity of the two algorithms (3.1 and 4.1) is also tested by conducting these two experiments. As shown in Table 5.1, p_{OL} and p_{OF} are the numbers of objects in the original decision rules of the leader and the follower, respectively (Refer to Step 1 of Algorithm 3.1), m_L and m_F are the condition attribute numbers of the leader and the follower, respectively, n_{OL} and n_{OF} are the numbers of the rules in the reduced objective decision rule sets of the leader and the follower, respectively, t_1 and t_2 are the processing times of Algorithms 3.1 and 4.1, respectively.

From the results shown in Table 5.1 we find that

1) The modelling algorithm (Alg. 3.1) and the solution algorithm (Alg. 4.1) are effective;

2) The processing time of Alg. 3.1 highly relates to the numbers of the rules in the original objective decision rule sets and the condition attribute numbers of the leader and the follower, respectively, expressed by the symbols p_{OL}, p_{OF}, m_L and m_F;

3) The processing time of Alg. 4.1 highly relates to the numbers of the rules in the reduced objective decision rule sets and the condition attribute numbers of the leader and the follower, respectively, expressed by n_{OL}, n_{OF}, m_L and m_F;

4）In the case that possible values of attributes in rule sets are few, such as LENSES, HAYES-ROTH, AUTO-MPG, PROCESSED_CLEVELAND, and BREAST-CANCER-WISCONSIN, Alg.4.1 is more efficient than the transformation-based solution algorithm [27].

VI. Conclusions and Future Study

Organizational decisions often require compromised solutions between two entities, but within guidelines determined by the decision entity at the upper level. Although bi-level programming has been widely studied, it can only be applied when a bi-level decision problem is modeled by mathematical functions. However, many real-world bi-level decision problems cannot be formulated as linear or

non-linear mathematical programs because various uncertain factors exist. To tackle this issue, our study proposes a rule-sets based bi-level decision approach, which first models a bi-level decision problem by rule-sets, and then finds a solution to the rule-sets based bi-level decision model. This paper presents a new solution algorithm to find an optimized solution for bi-level decision problems using rule set techniques with better algorithm complexity. Experiments, and the case study, demonstrate that the proposed rule-sets based bi-level decision approach is very effective for modelling and solving bi-level decision problems, in particular, when these problems cannot be modeled by the bi-level programming approach.

As future research, we will develop rule-sets based approaches for more complex bi-level decision problems: multi-objective and multi-follower bi-level decision problems. We will explore more real-world applications of the proposed approach, and also improve the current bi-level decision support system developed to help real decision makers solve their bi-level decision problems effectively.

ACKNOWLEDGMENT

The work presented in this paper was partially supported by Australian Research Council (ARC) under discovery grants (DP0557154) (DP110103733), the National Science Foundation of China (60675010), Postdoctor Foundation of China (230-21-86), and National Basic Research Priorities Program (2007CB311004).

REFERENCES

1. Von Stackelberg H. *The theory of market economy*. Oxford University Press, Oxford, 1952.
2. Chen CI, Gruz JB. Stackelberg solution for two person games with biased information patterns. *IEEE Transactions On Automatic Control* 1972; ***AC-17***, 791-798.
3. Bialas W, Karwan M. On two-level optimization. *IEEE Trans Automatic Control* 1982; ***AC-26***, 211-214.
4. Bard JF, Falk JE. An explicit solution to the multi-level programming problem. *Computers & Operations Research* 1982; ***9***, 77-100.
5. Bard JF, Moore JT. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics* 1992; ***39***, 419-435.
6. Bard JF. *Practical bilevel optimization: algorithms and applications*. Kluwer Academic Publishers, USA, 1998.
7. Dempe S. *Foundations of Bilevel Programming*, Kluwere Academic Publishers, 2002.
8. Candler W, Townsley R. A linear two-level programming problem. *Computers and Operations Research* 1982; ***9***, 59-76.
9. Bialas W, Karwan M. Two-level linear programming. *Management Science* 1984; ***30***, 1004-1020.
10. Hansen P, Jaumard B, Savard G. An extended branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* 1992; ***13***, 1194-1217.
11. Lan KM, Wen UP, Shih HS, Lee ES. A hybrid neural network approach to bilevel programming problems. *Applied Mathematics Letters* 2007; ***20***(8), 880-884.
12. Calvete HI, Galé C, Mateo PM. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 2008; ***188***(1), 14-28.
13. Aiyoshi E, Shimizu K. Hierarchical decentralized systems and its new solution By A Barrier Method. *IEEE Transactions on Systems, Man, and Cybernetics* 1982; ***11*****:** 444-449.

14. White D, Anandalingam G. A penalty function approach for solving bi-Level linear programs. *Journal of Global Optimization* 1993; ***3***, 397-419.

15. Liang L, Sheng SH. The stability analysis of bilevel decision and its application. *Decision And Decision Support System* 1992; ***2***, 63-70.

16. Wang GM, Wang XJ, Wan ZP, Lv YB. A globally convergent algorithm for a class of bilevel nonlinear programming problem. *Applied Mathematics and Computation 2007;* ***188***(1), 166-172.

17. Lu J, Ma J, Zhang G, Zhu Y, Zeng X, Koehl L. Theme-based comprehensive evaluation in new product development using fuzzy hierarchical criteria group decision-making method. *IEEE Transactions on Industrial Electronics* 2011; ***58***(6), 2236-2246.

18. Zhang G, Zhang GL, Gao Y, Lu J. Competitive strategic bidding optimization in electricity markets using bi-level programming and swarm technique. *IEEE Transactions on Industrial Electronics* 2011; ***58***(6), 2138-2146.

19. Nassif L, Nogueira J, de Andrade F. Resource selection in grid: a taxonomy and a new system based on decision theory, case-based reasoning, and fine-grain policies. *Concurrency and Computation: Practice and Experience* 2009; ***21***(3), 337–355.

20. Jaimes A, Coello C. MRMOGA: a new parallel multi-objective evolutionary algorithm based on the use of multiple resolutions. *Concurrency and Computation: Practice and Experience* 2007; ***19***(4), 397–441.

21. Sakawa M, Nishizaki I. Interactive fuzzy programming for two-level nonconvex programming problems with fuzzy parameters through genetic algorithms. *Fuzzy Sets and Systems* 2002; ***127***, 185-197.

22. Sakawa M, Yauchi K. Interactive decision making for multiobjective nonconvex programming problems with fuzzy numbers through coevolutionary genetic algorithms. *Fuzzy Sets and Systems* 2000; ***114***, 151-165.

23. Zhang G, Lu J. Model and approach of fuzzy bilevel decision making for logistics planning problem. *Journal of Enterprise Information Management* 2006; ***20***, 178-197.

24. Ma J, Fan Z, Jiang Y, Mao J. An optimization approach to multiperson decision making based on different formats of preference information, *IEEE Transactions on Systems, Man and Cybernetics, Part A* 2006; **36**(5), 876-889.

25. Li H, Li X, Hu D, Hao T, Liu W, Chen X. Adaptation rule learning for case-based reasoning. *Concurrency and Computation: Practice and Experience* 2009; ***21***(5), 673-689

26. Zheng Z, Lu, J, Zhang G and He Q. Rule sets based bilevel decision model and algorithm, *Expert Systems With Applications* 2009; ***36***(1), 18-26.

27. Zhang G, Zheng Z, Lu J and He Q. An algorithm for solving rule sets based bilevel decision problems, *Computational Intelligence* 2011; ***27***(2), 235-259.

28. Pawlak Z. *Rough sets theoretical aspects of reasoning about data*. Kluwer Academic Publishers, Boston, 1991.

29. Pooch UW. Translation of decision tables. *ACM Computing Survey* 1974; ***6***, 125-151.

30. Lew A, Tamanaha D. Decision table programming and reliability. *Proceedings of 2nd international conference of software engineering,* San Francisco, 1976; 345-349.

31. Wang GY. Rough Set Theory and Knowledge Acquisition. Press of Xi'an University of Communications, Xi'an, 2001.

32. Yao YY, Yao JT. Granular computing as a basis for consistent classification problems. *Communications of Institute of Information and Computing Machinery:special issue of PAKDD'02 Workshop on Toward the Foundation of Data Mining*, 2002; 101-106.

33. Tarski A. The concept of truth in formalized language, logic semantics, metamathematics. Clarendon Press, Oxford, 1956.

34. Quinlan JR. Learning efficient classification procedures and their application to chess end-games. In J.S. Michalski, J.G. Carbonell, & T.M. Mirchell, *Machine Learning: An Artificial Intelligence Approach, 1*. Morgan Kaufmann, Palo Alto, CA, 1983; 463-482.

35. Apolloni B, Brega A, Malchiodi D, Palmas G, Zanaboni AM. Learning rule representations from data. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 2006; ***36***(*5*): 1010-1028.

36. Zheng Z, Wang GY. RRIA: a rough set and rule tree based incremental knowledge acquisition algorithm. *Fundamenta Informaticae* 2004; ***59,*** 299-313.

37. Agrawal R, Srikant R. Fast algorithms for mining association rules. *Proceedings of the 20th international conference of very large data bases*, 1994; 487-499.

38. Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD conference on management of data,* Washington, D.C. 1993; 207-216.

39. Santini S, Jain R. Similarity measures. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 1999; ***21***, 871-883.

40. Han J, Kamber M. *Data mining concepts and techniques*. Morgan Kaufmann Publishers, 2001.

41. Pawlak Z. Rough sets, *International Journal of Information and Computer Science* 1982; ***11***, 341-356.

42. Hu XH, Cercone N. Learning in relational database: a rough set approach. *Computational Intelligence* 1995; ***11***, 323-338.

43. Shan N, Ziarko W, Hamilton HJ, Cercone N. Using rough sets as tools for knowledge discovery. *First international conference on knowledge discovery and data mining,* Menlo Park, CA. 1995; 263-268.

44. Mollestad T, Skowron A. A rough set framework for data mining of propositional default rules. *The 9th international symposium on methodologies for intelligent system,* 1996; 448-457.

45. Skowron A, Polkowski L. *Rough sets in knowledge discovery*. Physica Verlag, Heidelberg. 1998.

46. Wang GY, He X. A self-learning model under uncertain condition. *Chinese Journal of SoftWare* 2003; ***14***, 1096-1102.

47. Kiak A. Rough set theory: a data mining tool for semiconductor manufacturing. *IEEE Transaction on Electronics Packaging Manufacturing* 2001; ***24***, 44-50.

48. Pawlak Z, Slowinski K. Rough classification of patients after highly selective vagotomy duodenal ulcer. *International Journal of Man-Machine Studies* 1986; ***24***, 413-433.

49. Carlin US, Komorowski J, Ohrn A. Rough set analysis of patients with suspected of acute appendicitis. *Proceedings of the 7th conference on information processing and management of uncertainty in knowledge-based systems,* Paris, France. 1998; 1528-1533.

50. ROSETTA: The ROSETTA Homepage, 2006. http://www.rosettaproject.org/.

51. Wang GY, Zheng Z, Zhang Y. RIDAS-a rough set based intelligent data analysis system. *The first international conference on machine learning and cybernetics*, 2002; 646-649.

52. Jan GB, Marcin SS, Jakub W. A new version of rough set exploration system. *Lecture Notes In Computer Science, 2475*, 2002; 397-404.

53. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD international conference management of data*, Dallas, 2000; 1-12.

54. Ziarko W, Cercone N, Hu X. Rule discovery from databases with decision matrices. *Lecture Notes In Computer Science, 1079*, 1996; 653-662.

Table 2.1 An information table

Object	**height**	**hair**	**eyes**	**Class**
o_1	short	blond	blue	+
o_2	short	blond	brown	-
o_3	tall	dark	blue	+
o_4	tall	dark	blue	-
o_5	tall	dark	blue	-
o_6	tall	blond	blue	+
o_7	tall	dark	brown	-
o_8	short	blond	brown	-

Table 3.1 Objective decision rule set of the leader

Age	**Edulevel**	**Seniority**	**Health**	**Grade**
Young	High	Middle	Good	2
Middle	High	Long	Middle	2
Young	Short	Short	Poor	4
Young	Middle	Middle	Middle	2
Middle	Middle	Short	Middle	3
Middle	Middle	Long	Middle	2
Old	High	Long	Middle	3
Young	Short	Middle	Poor	2
Middle	Short	Short	Middle	4
Old	Short	Middle	Poor	4
Middle	Short	Long	Good	3
Middle	Short	Long	Middle	2
Old	High	Middle	Poor	3
Old	High	Long	Good	2
Old	Short	Long	Good	4
Young	High	Long	Good	4
Young	Short	Long	Middle	3

Table 3.2 Objective decision table of the follower

Age	**Edulevel**	**Seniority**	**Health**	**Grade**
Young	High	Long	Good	2
Old	Short	Short	Good	4
Young	High	Short	Good	2
Old	High	Long	Middle	3
Young	Short	Long	Middle	4
Middle	High	Middle	Poor	3
Middle	Short	Short	Poor	4

Old	Short	Short	Poor	4
Old	High	Long	Good	2
Young	Short	Long	Good	2
Young	Short	Middle	Middle	3
Middle	Short	Middle	Good	3
Old	High	Long	Good	2
Middle	High	Long	Good	2
Middle	High	Short	Poor	4

Table 4.1 Comparison of time complexities

	Alg. 4.1	T-Alg.
Maximal time complexity	$O\left(\left(n_{oL}+n_{oF}\right)\left(n_{cL}+n_{cF}\right)\left(m_L+m_F\right)+n_{oL}^{\min}\cdot\prod_{i=1}^{m_L}d_{c_{Li}}\cdot\prod_{i=1}^{m_F}d_{c_{Fi}}+n_{oF}^{\min}\cdot\prod_{i=1}^{m_F}d_{c_{Fi}}\right)$	$O\left(n_{oL}n_{oF}\left(n_{cL}+n_{cF}\right)\left(m_L+m_F\right)\right)$
Minimal time complexity	$O\left(\left(n_{oL}+n_{oF}\right)\left(n_{cL}+n_{cF}\right)\left(m_L+m_F\right)\right)$	$O\left(n_{oL}n_{oF}\left(n_{cL}+n_{cF}\right)\left(m_L+m_F\right)\right)$

Table 5.1 Testing results of Algorithms 3.1, 4.1 and the transformation-based solution algorithm

Data Sets	p_{OL}	p_{OF}	m_L	m_F	n_{OL}	n_{OF}	Alg. 3.1	Alg. 4.1	T-Alg.	Mark*
							t_1(sec.)	t_2 (sec.)	t_2 (sec.)	
LENSES	12	12	2	3	6	3	<0.01	0.02	0.03	1
HAYES-ROTH	50	50	2	3	21	24	<0.01	0.06	0.09	1
AUTO-MPG	199	199	4	4	80	76	0.08	0.24	0.39	1
BUPA	172	172	3	3	159	126	0.06	6.29	3.10	1
PROCESSED_ CLEVELAND	151	151	6	7	115	127	0.28	4.31	5.20	1
BREAST-CANCER -WISCONSIN	349	349	5	5	47	47	0.51	0.18	0.63	1

*Note， if mark=1, the algorithm is effective on the corresponding data set.

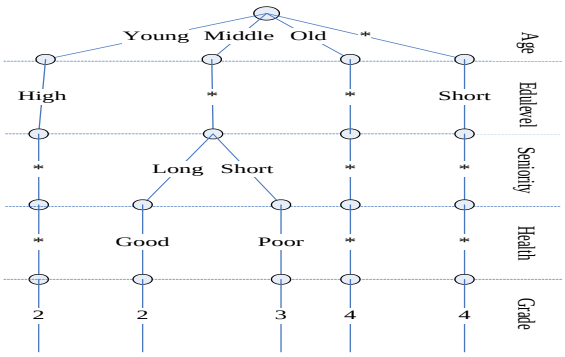

Fig. 2.1 An example of a rule tree

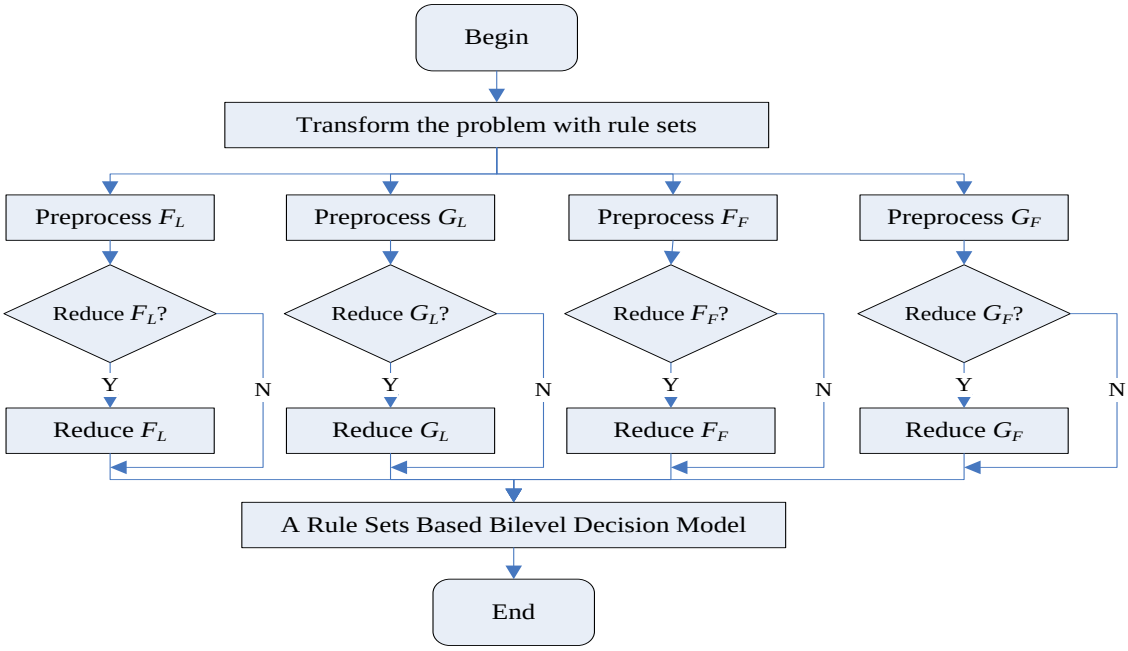

Fig. 3.1 Flow chart for the rule-sets based bilevel decision algorithm (Alg.3.1)

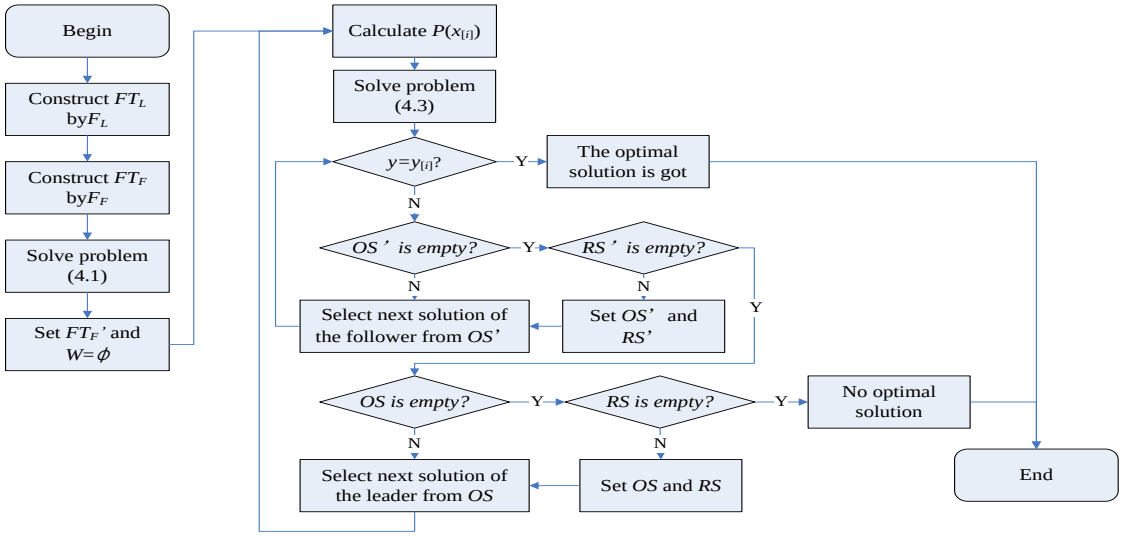

Fig. 4.1 Flow chart for the rule-sets based bilevel decision solution algorithm (Alg. 4.1)

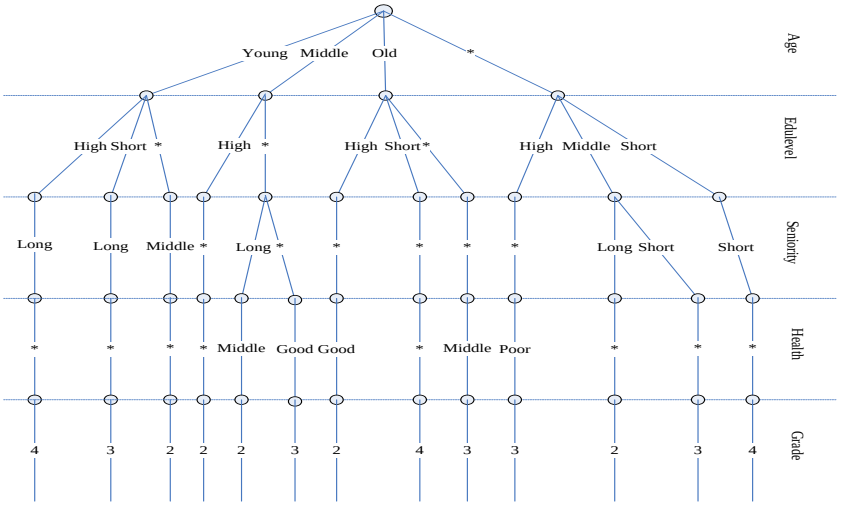

Fig. 4.2 Rule tree of the leader's objective decision rule set

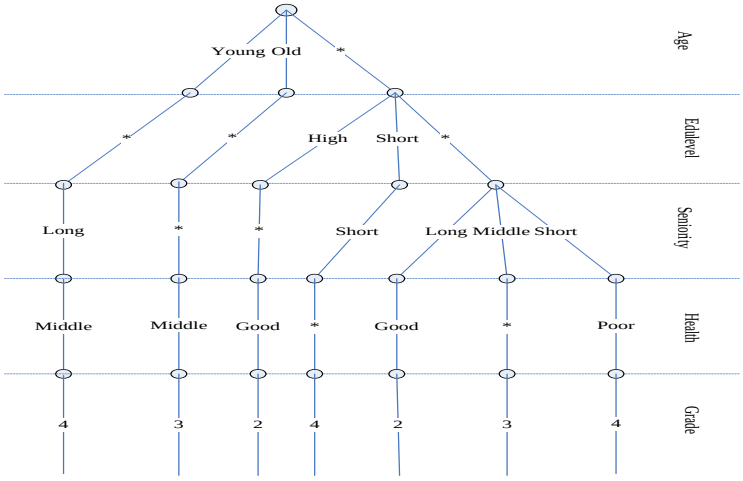

Fig. 4.3 Rule tree of the follower's objective decision rule set

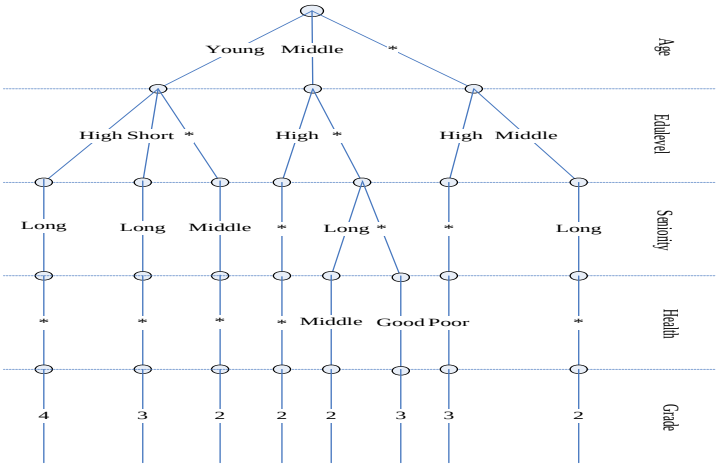

Fig. 4.4 Rule tree of the leader's objective decision rule set after cutting by constraint rule sets

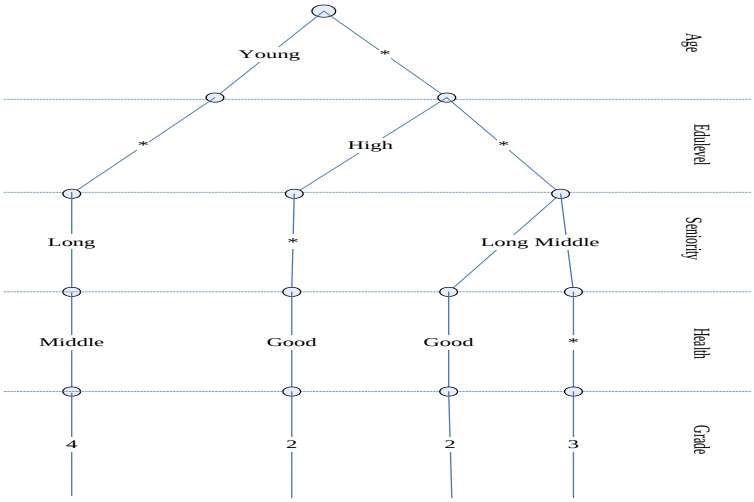

Fig. 4.5 Rule tree of the follower's objective decision rule set after cutting by constraint rule sets

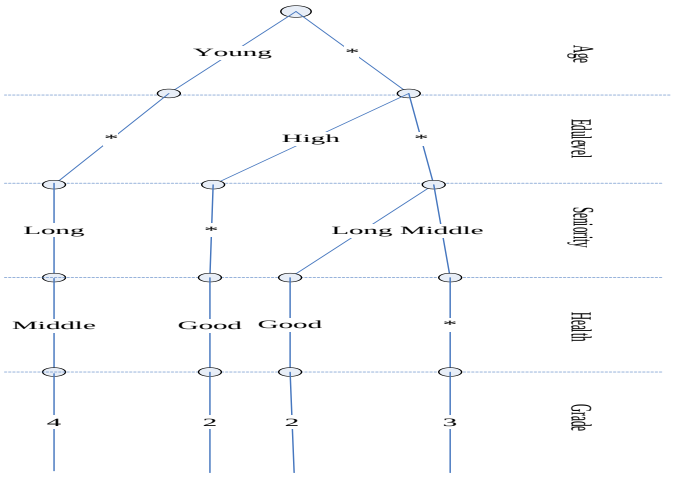

Fig. 4.6 Rule tree of the follower's objective decision rule set after pruning the rules that are not consistent with $x_{[i]}$