APPLICATION OF FIRST-ORDER RULES TO RECONSTRUCTING LINK DAMAGES IN LOGISTICS NET

S. Parshutin, G. Kuleshova and A. Borisov

Keywords: learning first-order rules, procedure FOIL, managing link damage

1. Introduction

Using logistic nets is an integral part of large-scale task management. Logistic nets are designed to be maximally stable; however, the risk of link damage of network elements cannot be excluded. Link damage reconstruction turns to be a specific and complicated process. Although logistic nets could be very much alike, they also possess individual properties – qualitative, probabilistic etc. One way to reconstruct links between the elements of the net is to employ first-order rules to describe a logistic net. This paper is concerned with the FOIL procedure (*First-Order Inductive Learner*) [1]. The execution of the procedure is shown for the particular logistic net. The rules obtained were used to search for alternative path in simulating the damage of a single or several links on the net.

2. Description of FOIL

Actually, FOIL is a system for finding function-free Horn clauses [1]. FOIL searches for first-order rules using a learning set. The search results in finding a set of logical rules describing the system under consideration [1, 2].

The first-order rule is a logical proposition of the form:

$$R(V_1, V_2, \ldots, V_k) \leftarrow L_1, L_2, \ldots, L_m, \quad (1)$$

where R is a target relation between variables V_i. In its turn, L_i are literals, composing a condition, whose satisfaction enables one to state that the head of the rule is true.

Information on possible relations between the objects observed in the system under consideration serves as the input data for FOIL. One of the relations (links) is considered to be the target relation. For each relation, tuples are composed, for which the link is true. The tuples contain constant values describing specific objects in the system. The data in the tuple is ranked according to the rank of variable $Q(V_i)$, which indicates the place in the tuple where the data for the variable is located. The tuples under which the relation is not true can also be specified for the target relation. Thus a learning set T is shaped. An assumption can also be made that none of the tuples except for the composed ones, is true. The tuples for which the link is true are called *positive*, whereas the others are treated as *negative* [1, 2].

An ideal variant for FOIL is to find a set of rules covering all positive tuples and none of negative. Figure 1 illustrates the FOIL algorithm.

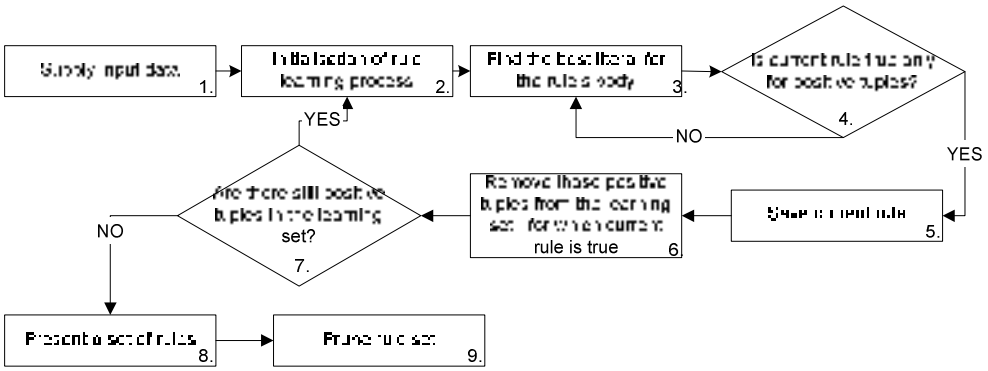

Figure 1. The FOIL algorithm

The algorithm starts by supplying the input data (Step 1). Also, a condition is specified under which the rule search must be stopped. For the present case it is condition: ”Does *T* contain positive tuples?” (Step 7). Besides that condition, the following rule search stopping criteria can be used: the number of iterations, procedure execution time, and the ratio of positive and negative tuples in the learning set [1, 2]. Then the process of rule search is initialised (Step 2). The target relation is placed in the head of the rule, and the rule assumes the following form:

$$R(V_1, V_2, \ldots, V_k) \leftarrow \ldots. \tag{2}$$

The next step is to search for the best literal for the rule’s body (Step 3), which is repeated until the rule becomes complete, i.e. true for positive tuples only (Step 4). The rule obtained is added to the current rule set (Step 5). After that, positive tuples covered by the added rule are removed from set *T*. (Step 6).

The process of rule search results in a set of rules (Step 8), which is then tested and pruned thus excluding logically unnecessary and inefficient rules (Step 9). The final set of first-order rules approximates a function set by the target relation and the input data [1,4].

Searching for literals for the rule’s body. FOIL uses the information gain based approach to find the best literal for the rule’s body. The next three formulas are used to evaluate a literal [2]:

$$I(T_i) = -\log_2\left(T_i^+ / \left(T_i^+ + T_i^-\right)\right), \tag{3}$$

$$I(T_{i+1}) = -\log_2\left(T_{i+1}^+ / \left(T_{i+1}^+ + T_{i+1}^-\right)\right), \tag{4}$$

$$Gain(L_j) = \left(T_i^+ - T_{i+1}^+\right) \times \left(I(T_i) - I(T_{i+1})\right), \tag{5}$$

where T_i^+ - positive tuples, T_i^- - negative tuples in the current learning set T_i ; T_{i+1}^+ and T_{i+1}^- describe learning set T_{i+1} after incorporating literal L_j in the rule’s body.

The literal with the best gain value is placed in the rule’s body provided that the rule’s body does not contain a literal identical to it (the same relation and the same variables). In the process of searching for the best literal, FOIL separates literals into *gainful* and *determinate*. A literal becomes determinate, if the gain is equal to zero. Other literals are considered gainful. In the case if all literals are determinate, a new variable have to be incorporated.

Incorporation of a new variable. A new variable is assigned the next in succession rank $Q(V_{i+1})$. A literal is then selected which will introduce a new variable, taking into account that at least one variable from those described previously must be present in the literal. Besides, the sum of ranks of variables in the literal with a single variable must be greater than the sum of ranks in any such literal introduced earlier. When the literal is selected, the number of elements in the tuples has to be increased so that the selected literal is true. The tuples whose

size cannot be increased, are removed from the learning set [1, 2]. In what follows an example is considered that includes the process of new variable incorporation.

3. Experimental part

To demonstrate the described technique, the simplest logistic net was used (see Figure 2). As regards the properties of the net, it is only known that two kinds of link exist between its elements: *linked_to(A,B)* – there is a direct link between elements *A* and *B*; *can_reach(A,B)* – there is an indirect link between elements *A* and *B*, assuming that using the direct links one can draw a path from *A* to *B*. The link *can_reach(A,B)* will also serve as a target link in the task under consideration.

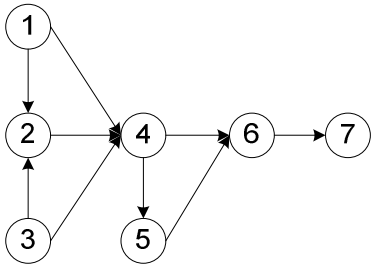

Figure 2. Logistic net

The essence of the task is to derive a set of rules describing the chosen logistic net and, in case of link damage, to find a roundabout path using the rules obtained.

3.1. Rule searching

Preparation of the input data. The links between the elements of the net are the following: *linked_to(A,B)* and *can_reach(A,B)*; the target relation is *can_reach(A,B)*.

Positive tuples for *linked_to(A,B)* are:

$$\{\langle 1,2\rangle,\langle 1,4\rangle,\langle 2,4\rangle,\langle 3,2\rangle,\langle 3,4\rangle,\langle 4,5\rangle,\langle 4,6\rangle,\langle 5,6\rangle,\langle 6,7\rangle\}.$$

The learning set T looks as follows:

- Positive tuples T^+ for *can_reach(A,B)* are:

$$\left\{\begin{array}{l}\langle 1,2\rangle,\langle 1,4\rangle,\langle 1,5\rangle,\langle 1,6\rangle,\langle 1,7\rangle,\langle 2,4\rangle,\langle 2,5\rangle,\langle 2,6\rangle,\langle 2,7\rangle,\langle 3,2\rangle,\\ \langle 3,4\rangle,\langle 3,5\rangle,\langle 3,6\rangle,\langle 3,7\rangle,\langle 4,5\rangle,\langle 4,6\rangle,\langle 4,7\rangle,\langle 5,6\rangle,\langle 5,7\rangle,\langle 6,7\rangle\end{array}\right\}$$

- Negative tuples for T^- *can_reach(A,B)* are:

$$\left\{\begin{array}{c}\langle 1,1\rangle,\langle 1,3\rangle,\langle 2,2\rangle,\langle 2,1\rangle,\langle 2,3\rangle,\langle 3,3\rangle,\langle 3,1\rangle,\langle 4,4\rangle,\langle 4,1\rangle,\langle 4,2\rangle,\langle 4,3\rangle,\langle 5,5\rangle,\langle 5,4\rangle,\langle 5,3\rangle,\langle 5,2\rangle,\langle 5,1\rangle,\\ \langle 6,6\rangle,\langle 6,5\rangle,\langle 6,4\rangle,\langle 6,3\rangle,\langle 6,2\rangle,\langle 6,1\rangle,\langle 7,7\rangle,\langle 7,6\rangle,\langle 7,5\rangle,\langle 7,4\rangle,\langle 7,3\rangle,\langle 7,2\rangle,\langle 7,1\rangle\end{array}\right\}$$

After the initialisation is completed, the searching process of literals for the rule's body starts. For each literal, the value of gain is calculated: $Gain(linked\_to) = 11.635$; $Gain(can\_reach) = 0$. The literal *linked_to(A,B)* is the only *gainful* literal, so it is included in the rule's body as being the best. The literal *can_reach(A,B)* is becoming *determinate*. Now the rule can be expresses as:

$$can\_reach(A,B) \leftarrow linked\_to(A,B).$$

The rule covers only positive tuples in set T; that is why it is considered complete. From set T, the tuples will be removed that are covered by the current rule, namely: $\{\langle 1,2\rangle,\langle 1,4\rangle,\langle 2,4\rangle,\langle 3,2\rangle,\langle 3,4\rangle,\langle 4,5\rangle,\langle 4,6\rangle,\langle 5,6\rangle,\langle 6,7\rangle\}$. The learning set T can be represented in this way:

- Positive tuples T^+ for *can_reach(A,B)*:
$$\{\langle 1,5\rangle,\langle 1,6\rangle,\langle 1,7\rangle,\langle 2,5\rangle,\langle 2,6\rangle,\langle 2,7\rangle,\langle 3,5\rangle,\langle 3,6\rangle,\langle 3,7\rangle,\langle 4,7\rangle,\langle 5,7\rangle\}$$
- Negative tuples for T^- *can_reach(A,B)*:

$$\left\{\begin{array}{c}\langle 1,1\rangle,\langle 1,3\rangle,\langle 2,2\rangle,\langle 2,1\rangle,\langle 2,3\rangle,\langle 3,3\rangle,\langle 3,1\rangle,\langle 4,4\rangle,\langle 4,1\rangle,\langle 4,2\rangle,\langle 4,3\rangle,\langle 5,5\rangle,\langle 5,4\rangle,\langle 5,3\rangle,\langle 5,2\rangle,\langle 5,1\rangle,\\ \langle 6,6\rangle,\langle 6,5\rangle,\langle 6,4\rangle,\langle 6,3\rangle,\langle 6,2\rangle,\langle 6,1\rangle,\langle 7,7\rangle,\langle 7,6\rangle,\langle 7,5\rangle,\langle 7,4\rangle,\langle 7,3\rangle,\langle 7,2\rangle,\langle 7,1\rangle\end{array}\right\}$$

The set T still contains positive tuples, due to that rule search has to be continued. The process of rule search is resumed, and the value of *Gain* is calculated for each gainful literal.

Literal *linked_to(A,B)* is becoming *determinate*, since $Gain(linked\_to)=0$. Furthermore, as *gainful* literals are missing, a decision is made to incorporate a new variable "C". The variable is assigned a rank. The rank for *A* is 1, for *B* is 2, so *C* is assigned the rank equal to 3. Now it is necessary to select a literal that will incorporate the new variable. Possible options are: *linked_to(A,C), linked_to(C,B), can_reach(A,C),* and *can_reach(C,B)*. The link *can_reach* is target, due to that it cannot be selected as the first literal in the rule's body. Let us assume, that the variable - if appeared for the first time-must take the place of the variable with the highest rank, that is literal *linked_to(A,C)* is selected.

The rule assumes the following form: $can\_reach(A,B) \leftarrow linked\_to(A,C)$.

The tuples will now look as $\langle A,B,C\rangle$. Before increasing the number of elements in the tuples of the current learning set, one has to remove the tuples whose size cannot be enlarged. In this case these are tuples in the form $\langle 7,\ldots\rangle$, since the tuples $\langle 7,\ldots\rangle$ are missing in the positive tuples for *linked_to(A,B)*. So, these tuples will be removed from the learning set T:

$$\{\langle 7,7\rangle,\langle 7,6\rangle,\langle 7,5\rangle,\langle 7,4\rangle,\langle 7,3\rangle,\langle 7,2\rangle,\langle 7,1\rangle\}.$$

Tuple extension will be performed according to the following algorithm:

$FOR\ \exists\langle X,Y\rangle \in T^+\ for\ linked\_to\ \ AND\ \exists\langle A,B\rangle \in T\ \ DO\ \ IF\ \ X=A\ \ THEN\ \ C=Y$.

The increased set will assume this form:

Positive tuples T^+ for *can_reach(A,B)*:

$$\left\{\begin{array}{c}\langle 1,5,2\rangle,\langle 1,5,4\rangle,\langle 1,6,2\rangle,\langle 1,6,4\rangle,\langle 1,7,2\rangle,\langle 1,7,4\rangle,\langle 2,5,4\rangle,\langle 2,6,4\rangle,\langle 2,7,4\rangle,\\ \langle 3,5,2\rangle,\langle 3,5,4\rangle,\langle 3,6,2\rangle,\langle 3,6,4\rangle,\langle 3,7,2\rangle,\langle 3,7,4\rangle,\langle 4,7,5\rangle,\langle 4,7,6\rangle,\langle 5,7,6\rangle\end{array}\right\}.$$

Negative tuples T^- for *can_reach(A,B)*:

$$\left\{\begin{array}{c}\langle 1,1,2\rangle,\langle 1,1,4\rangle,\langle 1,3,2\rangle,\langle 1,3,4\rangle,\langle 2,2,4\rangle,\langle 2,1,4\rangle,\langle 2,3,4\rangle,\langle 3,3,2\rangle,\langle 3,3,4\rangle,\langle 3,1,2\rangle,\\ \langle 3,1,4\rangle,\langle 4,4,5\rangle,\langle 4,4,6\rangle,\langle 4,1,5\rangle,\langle 4,1,6\rangle,\langle 4,2,5\rangle,\langle 4,2,6\rangle,\langle 4,3,5\rangle,\langle 4,3,6\rangle,\langle 5,5,6\rangle,\\ \langle 5,4,6\rangle,\langle 5,3,6\rangle,\langle 5,2,6\rangle,\langle 5,1,6\rangle,\langle 6,6,7\rangle,\langle 6,5,7\rangle,\langle 6,4,7\rangle,\langle 6,3,7\rangle,\langle 6,2,7\rangle,\langle 6,1,7\rangle\end{array}\right\}.$$

The current rule covers both positive and negative tuples in set T. Therefore it is considered incomplete, and searching for a next literal will start.

Possible variants are the following:

For *linked_to*:

1. *linked_to(A,B)* cannot be used as the sum of the ranks is less than in *linked_to(A,C)*; besides, the inclusion of this literal will lead to already existing rule;
2. *linked_to(A,C)* cannot be employed because this literal is already present in the rule;
3. *linked_to(C,B)* can be used. $Gain(linked\_to(C,B))=11.32$.

For *can_reach* we have:

1. *can_reach(A,B)* cannot be used as this literal is already present in the rule;
2. *can_reach(A,C)* can be used. $Gain(can\_reach(A,C)) = -\infty$;
3. *can_reach(C,B)* can be used. $Gain(can\_reach(C,B)) = 25.471$;

The performed analysis of possible variants has shown that literal *can_reach(C,B)* is the best and will be included in the rule's body. The rule assumes the following form:

$$can\_reach(A,B) \leftarrow linked\_to(A,C),\ can\_reach(C,B).$$

The rule is covering only positive tuples, therefore it is complete. All positive tuples will be removed from the learning set; due to that, the search for new rules will be stopped. The final set of rules contains two rules:

$$\left\{\begin{array}{l} can\_reach(A,B) \leftarrow linked\_to(A,B) \\ can\_reach(A,B) \leftarrow linked\_to(A,C),\ can\_reach(C,B) \end{array}\right\}.$$

Provided that the possibility of recursion is excluded at all, the final set of rules can be expressed as follows:

$$\left\{\begin{array}{l} can\_reach(A,B) \leftarrow linked\_to(A,B) \\ can\_reach(A,B) \leftarrow linked\_to(A,C),\ linked\_to(C,B) \\ can\_reach(A,B) \leftarrow linked\_to(A,C),\ linked\_to(C,D),\ linked\_to(D,B) \\ can\_reach(A,B) \leftarrow linked\_to(A,C),\ linked\_to(C,D),\ linked\_to(D,E),\ linked\_to(E,B) \\ can\_reach(A,B) \leftarrow linked\_to(A,C),\ linked\_to(C,D),\ linked\_to(D,E),\ linked\_to(E,F),\ linked\_to(F,B) \end{array}\right\}.$$

The set of rules describing a supply network (see Fig. 2) is obtained; so we can proceed to the second part of the task.

3.2. Reconstruing the damaged link

Single link damage. Assume that the system gives a signal that path *can_reach(1,7)* used at the present moment is damaged. System analysis is then conducted. As a result of direct link analysis, it is discovered that the link between elements "4" and "6" is damaged, namely, *linked_to(4,6)=false*. Hence, the path *can_reach(1,7)* must be reconstructed.

For *can_reach(1,7)*, this rule is satisfied:

$$can\_reach(1,7) \leftarrow linked\_to(1,4),\ can\_reach(4,7).$$

To retrace the complete path, we must go over to a rule without recursion. The following rule will be a rule of that kind:

$$can\_reach(1,7) \leftarrow linked\_to(1,4),\ linked\_to(4,6),\ linked\_to(6,7).$$

Now, an alternative to the damaged link has to be found; so it is necessary to check the following rule for truth:

$$can\_reach(1,7) \leftarrow linked\_to(1,4),\ can\_reach(4,6),\ linked\_to(6,7).$$

1. Rule $can\_reach(4,6) \leftarrow linked\_to(4,6)$ is false by the task statement;
2. Rule $can\_reach(4,6) \leftarrow linked\_to(4,C),\ can\_reach(C,6)$:
   a. Searching for the positive tuples in the form $\langle 4,C\rangle$ for *linked_to*: $\langle 4,C\rangle \in T^{+}(linked\_to): \{\langle 4,5\rangle\}$ must be performed;
   b. The rule assumes the following form:
   $$can\_reach(4,6) \leftarrow linked\_to(4,5),\ can\_reach(5,6).$$
3. The rule $can\_reach(5,6) \leftarrow linked\_to(5,6)$ is true; due to that, the recursive search can be stopped but the rule under testing can be considered true.
4. For the new path the rule $can\_reach(1,7) \leftarrow linked\_to(1,4),\ can\_reach(4,7)$ will also be true; however the path itself will look as follows:

$$can\_reach(1,7) \leftarrow linked\_to(1,4), linked\_to(4,5), linked\_to(5,6), linked\_to(6,7).$$

Damage of several links. In case if several damages of direct links are found, links are discovered in the system, which directly affect the integrity of the path, on whose damage the system reports. The elimination of unnecessary parts is performed taking into account that rule (a) which is correct for the current path, as well as the path itself optimal for the moment being, are already known to the system.

The damaged direct links are replaced with indirect ones, thus introducing a new rule-hypothesis, whose correctness has to be proved. At this moment an iterative process begins, when each of iterations repeats the process related to the reconstruction of a single link.

Example. Assume that the link *can_reach(1,7)* has to be reconstructed again. The current true rule is: $can\_reach(1,7) \leftarrow linked\_to(1,4), can\_reach(4,7)$, whereas the optimal path looks as follows: $can\_reach(1,7) \leftarrow linked\_to(1,4), linked\_to(4,6), linked\_to(6,7)$.

Analysis of the system has shown, that links *linked_to(1,4)*, *linked_to(5,6)* and *linked_to(6,7)* are not true anymore. The link *linked_to(5,6)* is not a part of the current path. Due to that, it will be removed; however, the lack of that link will not be forgotten as information of that kind affects searching for alternative path.

The rule for check can be expressed as

$$can\_reach(1,7) \leftarrow can\_reach(1,4), linked\_to(4,6), can\_reach(6,7).$$

For link $can\_reach(1,4)$ this path will be found:

$$can\_reach(1,4) \leftarrow linked\_to(1,2), linked\_to(2,4).$$

At the second iteration the following will occur. When checking the validity of the rule $can\_reach(6,7) \leftarrow linked\_to(6,C), can\_reach(C,7)$, we will find out that $\langle 4, C \rangle \in T^{+}(linked\_to): \{\ \}$. It disproves correctness of link $can\_reach(6,7)$, as well as the correctness of the rule tested.

As a result, the system gives signals of the lack of alternative path for the link *can_reach(1,7)*.

4. Conclusions

Searching for a possibility of discovering and reconstructing the damaged link on time turns to be a complicated task in the logistic net. This task will become even more complicated as the nets will grow. One way to solve the problem is to employ methods that approximate the function specified by the logistic net.

FOIL, whose algorithm is examined and discussed in this paper, is one of such techniques. The execution of the algorithm yields a set of first-order rules which approximate the function specified by the logistic net and by initial data available.

The results of the second part of the experiments show that the systems based on finding first-order rules can be used to solve the task of link damage in logistic net. There is no information on possible properties of the net itself, so the results only partly cover the problem of link reconstruction in the net.

The FOIL algorithm is not the final version of the method. The applied version does not foresee using functions in the rule; nevertheless this possibility is not excluded [3]. Also, the optimisation of the process of searching for literals for the rule's body and pruning final set of rules can improve the method under consideration. This paper focuses on the possibility of using first-order rules for link reconstruction in the logistic net and provides the ground for further research.

References

1. J.R. Quinlan and R.M. Cameron-Jones (1993). *FOIL: A Midterm Report*. Basser Department of Computer Science, University of Sydney.
2. J.R. Quinlan (1990). Learning Logical Definitions from Relations. *Machine Learning*, *5, 239-266 (1990)*.
3. J.R. Quinlan (1996). Learning First-Order Definitions of Functions. *Journal of Artificial Intelligence Research 5 (1996),* 139-161.
4. Tom. M. Mitchell (1997). *Machine Learning*. Chapter 10 – Learning sets of rules, 274-306.

Serge Parshutin, B.sc.ing., MSc. Student, Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: serge.parshutin@gmail.com

Galina Kuleshova, Mg.sc.ing., Research Assistant, Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: gk@egle.cs.rtu.lv

Arkady Borisov, Prof., dr.habil.sc.comp., Institute of Information Technology, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: aborisov@egle.cs.rtu.lv

Paršutins Sergejs, Kuļešova Gaļina, Borisovs Arkādijs. Pirmās pakāpes likumu izmantošana loģistikas tīkla saišu atjaunošanai
Loģistikas tīklu pielietošana ir liela mēroga uzdevumu vadīšanas neatņemama daļa. Tīkli tiek projektēti maksimāli stabili, tomēr nevienā loģistikas tīklā nevar izslēgt saišu bojājuma risku. Iespēja ātri atjaunot bojāto saiti ir viens no lielās prioritātes uzdevumiem. Šajā raksta ir izskatīta iespēja izmantot pirmās pakāpes likumus šī uzdevuma risināšanai. Likumu meklēšana notiek ar FOIL (First-Order Inductive Learner) algoritma palīdzību. Rezultātā ir iegūta likumu kopa, kas daļēji apraksta izmantoto tīklu pamatojoties uz esošo informāciju par tīkla īpašībām. Eksperimenta gaitā tika mākslīgi veidoti viena vai vairāku saišu bojājumi loģistikas tīklā. Uz FOIL algoritmu balstīts izstrādātais modelis tika veiksmīgi izmantots alternatīvu ceļu meklēšanai.

Parshutin Serge, Kuleshova Galina, Borisov Arkady. Application of first-order rules to reconstructing link damages in logistics net
The application of logistic nets is an integral part of large-scale tasks management. The nets are designed to be maximally stable; however, the risk of link damage between the elements of the net cannot be excluded. The possibility of reconstructing the damaged link on time, thus minimising the losses, is a task of high priority. This paper discusses using first-order rules for solving that task. To accomplish searching for first-order rules, the FOIL (First-Order Inductive Learner) algorithm is studied and employed. As a result, a set of first-order rules is obtained that partly describes the net used, based on the information about the properties of the logistic net. In the course of the experiment, damages of one or several links in the logistic network were created artificially. A model of the net constructed with the FOIL algorithm, was successfully applied to searching for alternative paths.

Паршутин Сергей, Кулешова Галина, Борисов Аркадий. Использование правил первого порядка для восстановления связей в логистической сети
Использование логистических сетей является неотъемлемой частью управления масштабными задачами. Сети проектируются максимально стабильными;однако но ни для одной из логистических сетей не исключён риск повреждения связи между элементами сети. Возможность быстро восстановить нарушенную связь, тем самым минимизируя потери, является одной из приоритетных задач. В данной работе рассмотрена возможность применения правил первого порядка для решения данной задачи. Для поиска правил первого порядка изучен и применён алгоритм FOIL (First-Order Inductive Learner). В результате получено множество правил первого порядка, частично описывающее использованную сеть, основываясь на предоставленной информации о свойствах логистической сети. В ходе эксперимента искусственно создавались повреждения одной или нескольких связей в логистической сети. Созданная с помощью FOIL модель сети эффективно применена для поиска альтернативных путей.