A Cooperative Multi-agent Data Mining Model and Its Application to Medical Data on Diabetes

Jie Gao[1], Jörg Denzinger[1], and Robert C. James[2]

[1] Department of Computer Science, University of Calgary, AB, Canada
{gaoj, denzinge}@cpsc.ucalgary.ca
[2] Aechidna Health Informatics, Winnipeg, MB, Canada
rob@aetiologic.ca

Abstract. We present *CoLe*, a model for cooperative agents for mining knowledge from heterogeneous data. *CoLe* allows for the cooperation of different mining agents and the combination of the mined knowledge into knowledge structures that no individual mining agent can produce alone. *CoLe* organizes the work in rounds so that knowledge discovered by one mining agent can help others in the next round. We implemented a multi-agent system based on *CoLe* for mining diabetes data, including an agent using a genetic algorithm for mining event sequences, an agent with improvements to the *PART* algorithm for our problem and a combination agent with methods to produce hybrid rules containing conjunctive and sequence conditions. In our experiments, the *CoLe*-based system outperformed the individual mining algorithms, with better rules and more rules of a certain quality. From the medical perspective, our system confirmed hypertension has a tight relation to diabetes, and it also suggested connections new to medical doctors.

1 Introduction

In recent years, data mining has been a hot topic that attracts both database and machine learning researchers. With the rapid development of data storage capacity, data mining calls for methods to handle large quantities of data, which could be heterogeneous with various types of data. However, most of the existing data mining methods are only capable of processing homogeneous data. Even many large-scale distributed data mining methods do not consider much about the heterogeneity of data. Another problem in data mining is that methods are more and more specialized (as was indicated in [1]). They perform well on ideal data sets. But when applied to real-world data, they often cause unsatisfactory results. Quite often different parts of a same data set are heterogeneous in characteristics. And a particular method may need different tuning for those parts.

Therefore it is necessary to propose a data mining model that allows for multiple different methods (or differently tuned methods) to work on one data set to handle its heterogeneity. This model should be a multi-agent system model so that we can use multiple agents to employ the different methods and consider

V. Gorodetsky, J. Liu, and V.A. Skormin (Eds.): AIS-ADM 2005, LNAI 3505, pp. 93–107, 2005.

the interaction and cooperation of these mining agents to produce integrated results as the knowledge discovered from the whole heterogeneous data set.

In this paper, we present a cooperative multi-agent data mining model, in which the agents use different methods to handle different types or parts of information in heterogeneous data sets. The results are combined to get integrated results. Critically, this model implies that agents cooperate with each other, so that our multi-agent mining model is not merely applying multiple algorithms to a data set, but trying to gain synergetic effects through the cooperation.

The rest of this paper is organized as follows: Our cooperative multi-agent mining model is introduced in Sect. 2. A concept-proving implementation in mining medical data is described in Sect. 3 and experimental results are presented in Sect. 4. Section 5 compares our model with some related work in distributed data mining. Finally Sect. 6 concludes this paper and potential work is given.

2 *CoLe*: Our Model for Cooperative Data Mining

To describe *CoLe* — our generic model of cooperative data mining, we need some basic definitions. First of all, our model works on different types of data. We can take them altogether as a "super data set". This super data set is called a *heterogeneous data set* (denoted by $\mathcal{D}$). The individual single-type data sets are called *simplex data sets* (denoted by $\mathcal{D}_i$) in this heterogeneous data set. Two simplex data sets of a heterogeneous data set do not necessarily need to be different. They are mainly used for showing the miners' ownerships (for *miners*, see below) of the simplex data sets. It is quite possible that the same simplex data set is worked on by several different mining algorithms.

From the data set we expect to discover *knowledge*. Although *knowledge* is an abstract concept and in many cases the knowledge representations depend on the specific mining method, we can represent (or convert) the knowledge in the majority of cases into an "if...then..." rule form. So we take a piece of knowledge as a *rule*, in the format of condition $\Rightarrow$ conclusion.

In our model, we have several mining methods employed to discover different rules from a given heterogeneous data set. For each simplex data set in it, we can have one or more suitable mining methods to discover knowledge from it. Such a mining method implementation is called a *miner*. It resembles a function $m : \mathcal{D}_i^* \rightarrow \mathcal{K}_i^*$, where $\mathcal{K}_i$ is the set of possible rules (knowledge) to be discovered by the miner. The rules expected to be discovered from different miners should have some conclusions in common (i.e. describing the same concept), so that we can combine the rule sets learned by different miners. Otherwise, the rules never have any conclusion in common, which gives us no hope to combine the rules.

In our cooperative multi-agent mining system, there are two types of agents. One type are miners, which implement the mining methods. The other type of agents are agents that combine the different types of rules. We assume there is only one such agent in a system (multiple such agents can be taken as a super agent). We call this agent *combination agent* ($\mathrm{Ag_{CBN}}$). It is at the core of the cooperation in our model.

Ag_{CBN} decides how the miners cooperate. Its main task is to receive the learned rule sets from individual miners, use some strategies to combine them, evaluate the combined rules against the whole heterogeneous data set and put the satisfactory ones into a final rule set, and at the same time extract helpful information from the rule sets and send it back to the miners so that each miner can utilize the discoveries from other miners.

The implementation of Ag_{CBN} can vary according to the different needs. It can be a simple agent that only validates the different combinations of the rule sets. It can also be a group of agents (forming a super Ag_{CBN}) to achieve complex interaction and cooperation strategies with the miners.

Basically our cooperative mining model contains the following elements: A heterogeneous data set $\mathcal{D}$ (whose simplex data sets decide what miners to use), a set of miners $\mathcal{M}$, and the combination agent Ag_{CBN}. And we name this model *CoLe* (*Co*operative *Le*arning):

$$CoLe =< \mathcal{D}, \mathcal{M}, \text{Ag}_{\text{CBN}} >$$

The agent interactions in *CoLe* are illustrated in Fig. 1: Given a heterogeneous data set $\mathcal{D}$, each miner m_i learns on the simplex data set ($\mathcal{D}_i$) it can handle. The mined set of rules ($\mathcal{R}_i$) are sent to Ag_{CBN}, which combines the rules and considers their quality in the whole heterogeneous data set $\mathcal{D}$, and puts the combination results to the final rule set $\mathcal{R}$. At the same time, for each miner m_i, Ag_{CBN} extracts useful information from $\mathcal{R}_1, \cdots, \mathcal{R}_{i-1}, \mathcal{R}_{i+1}, \cdots, \mathcal{R}_n$ and sends this feedback to m_i to help its work. The feedback could be rule condition fragments to help form good rules, data set clusters to help concentrate on uncovered data cases, etc.. In this model, the miners are cooperating via the intermediate Ag_{CBN}.

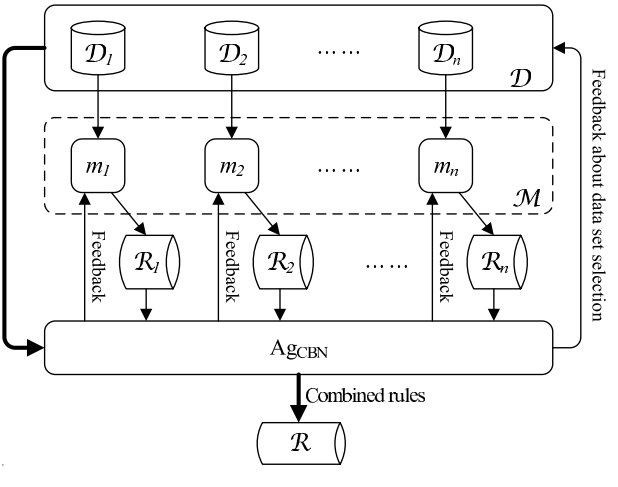

Fig. 1. *CoLe*model

In the above description, we propose a mine-and-feedback way of cooperation. In this way, the miners are synchronized to send their partial results after an iteration ends. $\mathrm{Ag_{CBN}}$ can then do its combination work and send feedbacks to help the miners' work in future iterations. The iterations continue until some end condition is met (e.g. an iteration number limit, or some rule quality threshold).

3 Application Problem and *CoLe* Solution

As a proof of concept for the *CoLe* model, a cooperative mining system was developed and implemented for mining diabetes patient data from the Calgary Health Region. This data set, originally used for public health insurance and billing purpose, contains very complete medical records. To identify the real disease cases from such data, the medical experts are not only interested in the diagnoses the individuals ever had, but, more over, they are also interested in the chronic development of the diseases. The *CoLe* mining model, with the ability to learn from different types of data, is very promising for their needs.

3.1 Problem Definition

In Sect. 2, the general *CoLe* model has already been discussed. When we fit the application problem — discover knowledge from medical history for identifying future diabetics — into the *CoLe* model, we have the instantiation as follows.

The *heterogeneous data set* in this problem is the public health data, which contains two different *simplex data sets*. One simplex data set $\mathcal{D}_\mathrm{s}$ is the medical records of the individuals. These timestamped records can be interpreted as discrete temporal sequences. The other simplex data set $\mathcal{D}_\mathrm{c}$ is the non-temporal version of these medical records (with only boolean fields for each possible diagnosis) and their annual statistics of the number of medical services, together with the personal information.

For $\mathcal{D}_\mathrm{s}$, we have a *sequence miner* ($\mathrm{Ag_S}$) to work on it. $\mathrm{Ag_S}$ identifies key diagnosis sequences that are the indication of future diabetics. These key diagnoses are called *events*. We only care about the relative order of the events. $\mathrm{Ag_S}$'s results are actually rules that use temporal sequences as conditions and the conclusion is always "diabetes". Such rules are called *sequence rules*.

For $\mathcal{D}_\mathrm{c}$, the miner is called *conjunctive miner* ($\mathrm{Ag_C}$) because its results are rules with conjunctions of predicates as conditions. The predicates are in the form "attribute *rel_op* value", where *rel_op* is a relation operator such as "=", ">","<", and the order of these predicates in a rule is of no consequence. These rules are called *conjunctive rules*.

We use a single combination agent $\mathrm{Ag_{CBN}}$ to combine the rules from $\mathrm{Ag_C}$ and $\mathrm{Ag_S}$ respectively. The combined rules may contain both sequence conditions and conjunctive conditions. Thus we call them *hybrid rules*.

In summary, in this problem of mining diabetes data, we have the data set $\mathcal{D} = \mathcal{D}_\mathrm{s} \bowtie \mathcal{D}_\mathrm{c}$, three types of rules: sequence rules, conjunctive rules and hybrid rules, the miner set $\mathcal{M} = \{\mathrm{Ag_S}, \mathrm{Ag_C}\}$, and the combination agent $\mathrm{Ag_{CBN}}$.

3.2 System Design

As already stated, there are three major components, $\mathrm{Ag_S}$, $\mathrm{Ag_C}$ and $\mathrm{Ag_{CBN}}$ in our system. Each miner receives its simplex data set, and learns rules from it. Both types of rules are sent to $\mathrm{Ag_{CBN}}$, which combines them into hybrid rules and then validates them against the heterogeneous data set. The hybrid rules with higher quality than a pre-defined threshold are put into the hybrid rule set as the result of the combination. At the same time, $\mathrm{Ag_{CBN}}$ also gives help to the two miners — generate the simplex data sets for the next iteration and send useful information extracted from $\mathrm{Ag_C}$'s results to $\mathrm{Ag_S}$ to help its future work. Such mining iterations will continue until our predefined iteration number limit has been reached.

Data Sets. The data set our system is going to work on is considerably large. To make the miners run efficiently, we do not let the miners run directly on the whole data set. Instead, we introduce *working data sets* with a smaller size for the miners to work on. In each of our mining iteration, $\mathrm{Ag_{CBN}}$ generates working data sets for $\mathrm{Ag_S}$ and $\mathrm{Ag_C}$. The working data sets contain the same set of instances but different types of data, fitting the two miners respectively. The details of generating the working data sets are presented in Ag_{CBM}'s design.

Sequence Miner. The sequence miner ($\mathrm{Ag_S}$) is the agent that does mining on the temporal simplex data set, and finds the sequence rules. $\mathrm{Ag_S}$ uses a genetic algorithm to achieve temporal rule mining. We designed a Michigan-like approach (see [2]) with sequence rules as individuals. A Michigan-like approach has a smaller granularity and its average run time is shorter than that of Pittsburgh approaches. This is suitable for dividing the whole genetic algorithm process into many short runs — more suitable for cooperation as required by *CoLe*.

Besides the commonly used genetic operators crossover and mutation, we designed an "intelligent" genetic operator called *intellicut* to make the evolutionary process more targeted. In *intellicut*, each event in the middle of an individual is checked to see if the quality of the individual can be increased by cutting off the events after this point, i.e., it will cut off a bad "tail" in the sequence. In this way, even if an individual is not very good, the good parts of it are preserved.

The fitness of an individual is evaluated according to its *accuracy* — the ratio of true positives to all instances it matches, and *coverage* — the ratio of true positives to all cases in the data set. These two factors are both considered to make sure the individuals contain valid knowledge and they will not overfit the data. We propose the following equation to calculate the fitness:

$$fitness = 10 \times \left(\frac{tp}{tp + fp}\right)^{x} \times \frac{\ln(tp)}{\ln(case\_num)} \tag{1}$$

In (1), tp and fp are the true positives (*cases* matched by this individual) and the false positives (*controls* matched by this individual) respectively, and *case_num* is the number of all *cases* in the data set. Here in the coverage calculation, the logarithm calculation makes the fitness less sensitive to coverage

when the coverage is big enough. The fitness is the product of the two factors so that neither the accuracy nor the coverage can be low. The exponent x of the accuracy controls the weight of the two factors in the fitness. In fact, (1) is used as a global assessment of rule quality throughout the system. Different x values are chosen for specific situations based on our experiments with various x values in these situations.

Conjunctive Miner. The conjunctive miner ($\mathrm{Ag_C}$) is the agent that mines on the non-temporal simplex data set to discover conjunctive rules.

The base mining algorithm used for $\mathrm{Ag_C}$ is the *PART* algorithm (see [3]). It forms rules from pruned partial decision trees built using *C4.5*'s learning method (for *C4.5*, see [4]). The resulting rules are in the form of conjunctive rules. *PART* has no global optimization and therefore can return a rule immediately after discovering it. This gives us the ability to interrupt it halfway with partial results, allowing run time limits for $\mathrm{Ag_C}$ and easy synchronization with other miners. In our system we use an implementation of *PART* directly from a machine learning package called *WEKA* (see [5]).

As *PART* is designed for creating rule sets for classification problems, it is necessary to do some pre- and post-mining work to make the results more suitable for our cooperative mining problem. The pre-mining tasks are mainly used for data reduction. We use a *relevance factor*:

$$RF(A) = \Pr(A) \times \log\left(\frac{\Pr(A|case)}{\Pr(A|control)}\right) \tag{2}$$

(inspired by [6]) to identify relevant possible diagnoses and eliminate irrelevant ones to reduce the number of attributes. The post-mining task is to generalize the results — generate simpler rules based on the results and use fitter (measured by (1)) ones in the simpler rules to replace the original ones.

Subrule Checking in Both Miners. Subrule checking is done in both $\mathrm{Ag_S}$ and $\mathrm{Ag_C}$ before sending their results to $\mathrm{Ag_{CBN}}$ to prepare more candidates for combination. A subrule is one whose condition is a subset of the original condition (and for sequence rules, the events should also appear in the same relative order as the original rule). For each rule, all the subrules of this rule will be checked against a fitness threshold over the entire data set $\mathcal{D}$. All qualified subrules are sent together with the original result set to $\mathrm{Ag_{CBN}}$ as the result of an individual miner. This brings more "materials" for the combination while the earlier tasks increase the overall rule quality.

Combination Agent. The cooperation is mainly achieved by $\mathrm{Ag_{CBN}}$ combining the results from the miners to produce hybrid rules.

In $\mathrm{Ag_{CBN}}$, the combination inputs are *srules* and *crules*, which are the result rule sets of $\mathrm{Ag_S}$ and $\mathrm{Ag_C}$, respectively. The combination is done in several stages:

1. Direct combination: Combine a sequence rule with a conjunctive rule (including subrules)

2. Crossing combination: Convert a predicate to an event (or vise versa) and combine with existing hybrid rules
3. Rule pruning: Remove duplicates and unnecessary parts
4. Working data set generation: Generate the working data set for the next iteration to the miners
5. Hints to $\mathrm{Ag_S}$: Give hints to $\mathrm{Ag_S}$ according to $\mathrm{Ag_C}$'s results.

In these stages, a rule's quality is also measured by the fitness calculated by (1). A fitness threshold ft is set before hand. The combined rules with fitness greater than ft will be put into the hybrid rule set *hrules*. We assume all these rule sets contain only rules with a conclusion "diabetes", because the knowledge we intend to discover is "what a diabetes patient should be like" instead of "what a diabetes patient should not be like".

In direct combination, if a rule in *srules* or *crules* already has greater fitness than ft, it will be put to *hrules* directly. Then new hybrid rules are created by putting a sequence rule's condition and a conjunctive rule's together. The new hybrid rules with higher fitness than ft are put into *hrules* as well.

The second stage, crossing combination, uses the hybrid rules in *hrules* after the first stage, together with *srules* and *crules*, as the base. The basic idea of crossing combination is to convert some diagnostic predicates in conjunctive rules to events and put them into the sequence parts of a hybrid rule to see if the hybrid rule's fitness increases. A similar event-to-predicate combination is also done. The conversions can be made because both the diagnostic predicates and events are on the same set of possible diagnoses. For example, we can convert a predicate "Diagnosis_A = true" to an event "[Diagnosis_A]". In this way, the use of one miner's result to help the other is maximized. A hill-climbing method is used in the iterations to find the best results for each hybrid rule's combination.

After the combination stages, the resulting rule set *hrules* is pruned to eliminate duplicate rules and useless conditions. Single-event sequences are converted to a predicate since a single event cannot indicate any temporal order. A predicate that has a counterpart in the sequence part is erased because the condition in the sequence part is stronger than the conjunctive part. And finally duplicate rules are erased.

$\mathrm{Ag_{CBN}}$ then generates the working data sets for the two miners. The first working data set is generated randomly from the whole data set before the mining work starts. Later ones are generated according to the combination results, based on the previous working data set. We denote the previous working data set and the new (currently generating) one as W_o and W_n respectively. $\mathrm{Ag_{CBN}}$ first eliminates the correctly covered instances (true positives), as well as the true negatives covered by conjunctive rules predicting *controls*, from W_o. The remaining instances can take only up to 80% in W_n. The rest pending instances in W_n are randomly picked from the whole data set. By doing this, $\mathrm{Ag_{CBN}}$ can guide the miners to focus on the cases that have not been covered by existing rules without driving the miners into smaller and smaller corners.

The predicates of the conjunctive rules from $\mathrm{Ag_C}$ may contain some key indicators of diabetes. So they can act as domain specific knowledge to $\mathrm{Ag_S}$.

After the combination, Ag_{CBN} will extract the predicate attributes from Ag_C's result, and make some candidate sequence segments from them in order to help Ag_S's work. These candidate segments can be used by the mutation operator in Ag_S's genetic algorithm. For example (shown in Fig. 2), we have conjunctive rules cr_1 and cr_2 coming from the conjunctive rule miner. Let $evt_1, \cdots, evt_5$ correspond to $pre_1, \cdots, pre_5$. We have sequence segments $ss_1, ss_2, \cdots, ss_8$ to be used as candidate individual parts in Ag_S. We do not have a counterpart for giving hints to Ag_C, because Ag_C is using an existing implementation and we have few ways to influence the *PART* algorithm directly.

cr_1:$pre_1 \wedge pre_3 \Longrightarrow$ disease cr_2:$pre_2 \wedge pre_4 \wedge pre_5 \Longrightarrow$ disease

can generate sequence segments:

ss_1: $evt_1 \rightarrow evt_3$ ss_2: $evt_3 \rightarrow evt_1$ ss_3:$evt_2 \rightarrow evt_4 \rightarrow evt_5$

ss_4:$evt_2 \rightarrow evt_5 \rightarrow evt_4$ ss_5:$evt_4 \rightarrow evt_2 \rightarrow evt_5$ ss_6:$evt_4 \rightarrow evt_5 \rightarrow evt_2$

ss_7:$evt_5 \rightarrow evt_2 \rightarrow evt_4$ ss_8:$evt_5 \rightarrow evt_4 \rightarrow evt_2$

Fig. 2. Hints from Ag_C to Ag_S example

4 Experimental Evaluation

Our cooperative mining system has been tested with different experiments to show the advantage of cooperation.

4.1 Data Preparation

The diabetes medical control data for our mining system comes from the Calgary Health Region. The data contains population born before 1954 and have been living in Calgary continuously since 1994. We want to analyze the medical records 5 years prior to the identification of diabetes. So we keep only the individuals who have no diagnoses of diabetes before 2000 but have at least one diabetes diagnosis in 2000, i.e., first diagnosed as diabetes patients in 2000. They are the *cases* in our data. For each of the *cases*, we also give 2 *controls* who are in the same sex and similar age but have no diabetes diagnoses at all.

In the original data set, there are three tables. One is the registration table (REG table) containing ID, age, gender, and class (*case* or *control*). The other two are medical records, one for hospital (HOSP table) and the other for clinical services (MD table). The medical records are mainly the diagnostic codes given by the doctors, together with the date of service. In Table 1 the sample data tables are shown (some unused fields are omitted). We put the clinical and hospital medical records together and only extract the records from 1995 to 1999 — the 5-year period before diabetes diagnoses.

The diagnostic codes are defined by the International Classification of Diseases, 9^{th} revision (ICD-9, see [7]). We use a higher-level abstraction, the ICD-9 Basic Tabulation List ([8]), to aggregate the diagnostic codes to 307 disease

Table 1. Sample data tables

REG

ID	CC	GENDER	YEAR
2	0	M	1921
3	0	F	1922
5	0	F	1946
6	0	F	1930
7	1	M	1938
18	1	F	1950
19	1	M	1940
...	...		

HOSP

ADMIT	DX_1	DX_2	DX_3	DX_4	DX_5	...	ID
1996-12-21	9975	5968	E8788	7140	36610	...	3
1997-06-19	57420					...	3
1997-06-30	57410	V1301				...	3
1998-04-08	9962	E8781				...	7
1999-04-16	99677	72709	2851	E8781		...	7
1999-08-12	5409					...	7
1998-04-27	9962	99813	E8781			...	7
... ...							

MD

...	SERV_SDATE	DIAG1	DIAG2	DIAG3	ID
...	2000-03-06				2
...	2000-03-25	595			2
...	1997-06-27	594.9	788.0		3
...	1995-12-07	466			5
...	1999-12-14	733	717.8	719.4	5
...	1999-10-08	174			5
...	2000-02-25	780.5			6
... ...					

types in about 70 categories, to condense the data. Then we split the data set into sequence and conjunctive simplex data sets. For the sequence simplex data set, we order each instance's diagnoses, both hospital and clinical, by the date of service. For the conjunctive simplex data set, we put the basic information of the instances — age, gender, etc. — together with a boolean table for the diagnoses. If an instance ever had a diagnosis, the corresponding field has the value *true*, and vice versa.

There were two data sets prepared for the tests. A small data set contains 1800 instances, with 600 *cases* and 1200 *controls*. The corresponding working data set for the miners contains 900 instances. This small data set is used for most of the tests and comparisons in Sect. 4.2. We also prepared a large data set with all valid instances we have, containing 9450 instances (3150 *cases* and 6300 *controls*). The corresponding working data set size is 3150 instances. Test runs on this larger data set were made to get rules for evaluation of the knowledge discoveries in Sect. 4.3.

4.2 The Effect of Combination

Our first evaluation is the comparison of the single miners and the combinations so that we can show the effect of combination in our system. We had five test runs with different fitness thresholds in $\mathrm{Ag_{CBN}}$. These fitness thresholds control how a rule is qualified (see Sect. 3.2). Other parameter values were decided by some

experiments before hand so that their values used in the tests were reasonable and the same for all our tests here. Results of these tests are shown in Table 2. The run time for each of the tests was within 15 hours, which is quite acceptable. As our concentration is on the cooperation and combination, we do not put too much focus on the run time.

Table 2. Tests with different thresholds

Test No.		1	2	3	4	5
Fitness threshold		3.8	3.9	4.0	4.1	4.2
Rule origins	Ag_S	263	104	25	13	5
	Ag_C	0	0	3	0	0
	Combination	5745	1852	743	287	72
	Total	6008	1956	771	300	77
Fitness averages	Ag_S	3.973	4.089	4.162	4.282	4.251
	Ag_C	N/A	N/A	4.183	N/A	N/A
	Combination	4.054	4.117	4.258	4.312	4.344
	Overall	4.051	4.116	4.254	4.311	4.338
True positive averages	Ag_S	67.18	61.73	65.36	26.62	33.00
	Ag_C	N/A	N/A	46.333	N/A	N/A
	Combination	56.55	55.66	45.92	26.49	25.25
	Overall	57.02	55.98	46.55	26.49	25.75

The first comparison is the number of rules with different origins. Ag_S or Ag_C-origin rules include both qualified original rules from a miner and the qualified subrules in subrule checking. The rules with contributions from both miners take "Combination" as their origins. When our system records the origins, single-miner origins have higher priority, i.e. when a rule discovered by a single miner happens to be re-discovered by Ag_{CBN}, we give credit to the single miner instead of Ag_{CBN}. In Table 2 we can clearly see that the combined rules are much more than the ones with origins Ag_S or Ag_C. This is the first indication that the combination can produce more potential good rules according to our fitness measure.

The average fitness of the rules by origin are calculated as well. The averages differ as the thresholds are different. The average fitness of the combined rules is a bit higher than the ones from single miners. This again indicates the combination increases the quality of the discovered knowledge, or in another word, produce better knowledge from two types of materials.

In each test run the combined rules have comparable true positive coverage with the ones from single miners. This shows that our combination strategies are not overfitting the given data. Although the true positive coverages of combined rules are a bit lower than the ones of individual miners, this shows that the com-

bined rules have higher accuracy, because our fitness measure in (1) is balanced between coverage and accuracy and the combined rules have higher fitness.

In addition to the summaries for the whole rule sets, summaries for the top 10 (by fitness) rules in each test are also presented here in Table 3. From this table we can see that most of the top 10 rules are combined rules. And in the only test with rules from Ag_S in the top 10, the combined rules have a higher average fitness than the ones from Ag_S. For the same reason indicated above, in Test No. 2 the true positive average of rules from Ag_S is a bit higher than that of the combined rules, indicating combined rules have higher accuracy.

Table 3. Tests with different thresholds (top 10 rules)

Test No.		1	2	3	4	5
Fitness threshold		3.8	3.9	4.0	4.1	4.2
Rule origins	Ag_S	0	2	0	0	0
	Ag_C	0	0	0	0	0
	Combination	10	8	10	10	10
	Total	Top 10				
Fitness averages	Ag_S	N/A	4.594	N/A	N/A	N/A
	Ag_C	N/A	N/A	N/A	N/A	N/A
	Combination	4.769	4.684	4.821	4.696	4.491
	Overall	4.769	4.666	4.821	4.696	4.491
True positive averages	Ag_S	N/A	24.50	N/A	N/A	N/A
	Ag_C	N/A	N/A	N/A	N/A	N/A
	Combination	23.80	23.38	24.40	23.20	25.60
	Overall	23.80	23.60	24.40	23.20	25.60

The use of hints to Ag_S was specially tested and evaluated to prove its contribution.

The tests were to run Ag_S's genetic algorithms with and without hints, to see if the results will be different in quality. To get some valid hints for the tests, the run log in test No. 3 in Table 2 was used and we extracted the hints and working data set in the tenth iteration from it. The tests made here all ran on this working data set, and the ones with hints used this hint set. To make sure we have a general result without the interference of random numbers, we repeat each test individually for five times. So totally we have 5 runs with hints, and 5 without hints. In each run, we use a generation size of 300, evolve the individuals for 50 generations, and finally filter out those individuals with fitness greater than 2.8.

Table 4 shows the qualified individual numbers for each test run. From the numbers and the averages, an obvious conclusion is that the runs with hints generate more qualified individuals. More detailed observations are made on the evolutionary process. In Fig. 3, average fitnesses of the first 10 generations in

each test run's evolutionary process are presented in a line chart. We can see the average fitnessses for test runs with hints increase much faster than the ones without hints.

Table 4. Number of qualified individuals

Generation size	300					
Generation limit	50					
Fitness threshold	2.8					
Test No.	1	2	3	4	5	Average
With hints	13	33	33	51	63	38.60
Without hints	7	9	12	15	28	14.20

From the results shown in Table 4 and Fig. 3, we can conclude that Ags does benefit from hints so that it can start faster and get more out of the data.

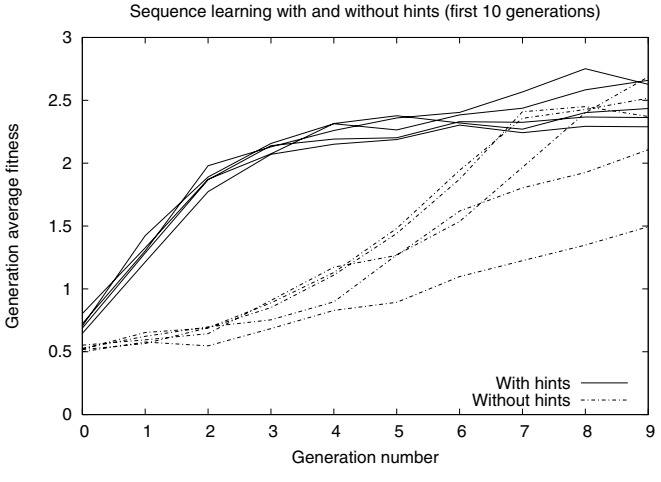

Fig. 3. Average fitness with and without hints (first 10 generations)

4.3 Knowledge Discovered

In this section the discoveries of our test runs on the full data set are examined and analyzed. We find in our results there are not only rules matching known facts about diabetes, but also promising and interesting discoveries that were new to the medical experts.

The most important discovery of our miner is the relation of hypertension and diabetes. In the medical field, it is already known that diabetes has a tight

relation with hypertensive diseases. In our test runs, 100% of the discovered hybrid rules have either "hypertensive diseases" in the sequence part as an indicative event, or "hypertensive diseases=*true*" in the conjunctive part. With such high occurrences of hypertensive diseases diagnoses in our results, it is a very exciting result to the medical experts.

Another discovery is about a general diagnose "signs, symptoms and ill-defined conditions". This diagnosis also has high occurrences in our results. However, unlike the hypertension diagnoses, this diagnosis cannot tell us about any specific diseases or disorders, but only some indication in general that the patient does not feel well. In particular, many rules come in the form like Fig. 4, where the "signs, symptoms and ill-defined conditions" diagnoses appear repeatedly in the sequence part. This is an indication that the patients may have been feeling uncomfortable for long before diabetes related diagnoses are made. Among all our *cases* there are about 25% who match the rule in Fig. 4.

Part	Condition (ICD-9)	Description
Conjunctive	{466,480-519}=1	Other diseases of the respiratory system
Sequence	{780-799}	Signs, symptoms and ill-defined conditions
	{780-799}	Signs, symptoms and ill-defined conditions
	{780-799}	Signs, symptoms and ill-defined conditions
	{401-405}	Hypertensive disease

Fig. 4. A hybrid rule

Another frequent diagnosis, "other diseases of the respiratory system", has an average of over 80% to appear in all the final hybrid rules (the rule in Fig. 4 is one of them). In medical doctors' eyes, this discovery does not have an obvious explanation (according to the experts we have talked to so far). This should be an interesting topic for the medical experts.

5 Related Work

According to the categories for cooperative search problems discussed in [9] (knowledge-based search is essentially the core in data mining methods), the *CoLe* model has characteristics from both *dividing the problem into subproblems* — split of the heterogeneous data set into simplex data sets and use of multiple miners — and *improving on the competition approach* — result segments from different miners are competing to appear in the combined rules. Depending on the way the heterogeneous data set is split and the strategies for combination in $\mathrm{Ag_{CBN}}$, the *CoLe* model's similarity to *dividing the problem into subproblems* and *improving on the competition approach* may vary.

Compared to existing applications, the *CoLe* model presents some new ideas and improvements.

In [10], a theorem proving system named *TECHS* is presented. Two types of provers are used, namely universal provers and specialized provers respectively. In the proving process, the provers exchange selected clauses periodically with each other and integrate received clauses into their own search states. Our *CoLe* model has some design similarities with *TECHS*. However, the major difference is that *CoLe*'s combination of rules is done in a central agent $\mathrm{Ag}_{\mathrm{CBN}}$, while in *TECHS* each agent is responsible for its own integration of others' results. The central combination agent in *CoLe* can have a global view on all the different types of results and thus has a better chance to gain good rules through combination.

Viktor et al. developed *CILT* (see [11]), in which several agents form the cooperative mining team. The agents are either *machine miners* or *human miners*. The machine miners each employ a different data mining technique to discover knowledge from data; and the human miners obtain knowledge from human experts. Although in *CILT* different algorithms and different types of miners are used, they all produce the same (compatible) type of results. This leads to few potentials of strengthening knowledge through combination, and knowledge cannot be represented in hybrid rule format, which fits the complex real world better. In *CoLe*, we consider the results from the miners to be heterogeneous. Combining these heterogeneous results can gain hybrid results that can not be achieved by any single miner.

Most importantly, the *CoLe* model emphasizes the combination of the different types of rules, which is not a key characteristic in any of the existing cooperative distributed methods. The combination is not only an effort to make the cooperation more thoroughly but it also enhances the knowledge that is discovered. This can lead to greater synergetic effects.

6 Conclusion and Future Work

We proposed a cooperative multi-agent mining model — *CoLe*. In *CoLe*, multiple miners use different algorithms to mine a heterogeneous data set. These results are sent to a combination agent to create hybrid results and extract useful information as feedbacks to the miners. *CoLe* highlights the interaction and cooperation of different algorithms and the combination of different-typed results from different algorithms, resulting in synergetic improvements against the single algorithms.

We implemented a mining system using our *CoLe* model, aimed at mining knowledge from diabetes data with both temporal and non-temporal information. We use a sequence miner and a conjunctive miner for mining. The combination agent in this system contains our strategies for combination and cooperation. Experiments showed clearly how our combined hybrid results enhance and strengthen the raw results from single miners. Additionally, in the medical field, these results not only confirmed known knowledge about diabetes, namely that

hypertension has a tight relation with diabetes, but also found some observations new to the medical experts.

The implemented mining system is the first proof of concept for our *CoLe* model. While it shows good results in the experiments, we have a lot of potential work to do. First, more implementations of cooperative mining systems using *CoLe* should be made to show that our *CoLe* model is a general paradigm for various learning/mining problems. On the conceptual side, more ways are needed to provide feedback from one miner (or the combination agent) to others. On the medical data mining side, future work should include: making use of other information together with diagnoses; mining on data sets where the distribution of disease cases is more close to general population; and mining data on more diseases.

References

1. Fayyad, U., Uthurusamy, R.: Evolving Data into Mining Solutions for Insights. Communications of the ACM **45** (2002) 28–31
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional (1989)
3. Frank, E., Witten, I.H.: Generating Accurate Rule Sets Without Global Optimization. In: Proceedings of the 15^{th} International Conference on Machine Learning, Morgan Kaufmann (1998) 144–151
4. Quinlan, J.R.: C4.5: Programs for Machine Learning. The Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann (1993)
5. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
6. Liu, H., Lu, H., Yao, J.: Toward Multidatabase Mining: Identifying Relevant Databases. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 541–553
7. Karaffa, M.C.: International Classification of Diseases, 9^{th} Revision, 4^{th} Edition, Clinical Modification. Practice Management Information Corp., Los Angeles (1992)
8. World Health Organization: International Classification of Diseases, 9^{th} Revision: Basic Tabulation List with Alphabetical Index. World Health Organization, Geneva (1978)
9. Denzinger, J.: Conflict Handling in Collaborative Search. In Tessier, Chaudron, Müller, eds.: Conflicting Agents: Conflict Management in Multi-agent Systems, Kluwer Academic Publishers (2000) 251–278
10. Denzinger, J., Fuchs, D.: Cooperation of Heterogeneous Provers. In: Proceedings of the 16^{th} International Joint Conference on Artificial Intelligence (IJCAI'99), Stockholm, Sweden, Morgan Kaufmann (1999)
11. Viktor, H.L., Arndt, H.: Data Mining in Practice: From Data to Knowledge Using a Hybrid Mining Approach. The International Journal of Computers, Systems and Signals. **1** (2000) 139–153