Таким образом, применение технологии когнитивного моделирования позволяет действовать на опережение и не доводить потенциально опасные ситуации до угрожающих и конфликтных, а в случае их возникновения – принимать рациональные решения в интересах субъектов России.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Исследование операций: В 2-х т. Пер. с англ. / Под ред. Дж. Моудера, С. Элмаграби.– М.: Мир, 1981. Т. 1.– 712 с.
2. Краткий словарь когнитивных терминов. Сост. Е.С. Кубрякова, В.З. Демьянков, Ю.Г. Панкрац, Л.Г. Лузина.– М., 1997.
3. *Меркулов И.П.* Когнитивная наука // Новая философская энциклопедия в четырех томах. Т. 2.– М., 2001.
4. *Монсон П.* Современная западная социология.– СПб, 1992.
5. *Сафронова В.М.* Прогнозирование и моделирование в социальной работе: Учеб. пособие для студ. высш. учеб. заведений.– М.: Издательский центр «Академия», 2002.– 192 с.
6. *Солсо Р.Л.* Когнитивная психология.– СПб.: Питер, 2002.– 592 с.
7. *Чернов И.В.* Имитационное моделирование при исследовании процессов, возникающих в социально-экономических системах // Институт проблем управления РАН, Москва.
8. Structure of Decision. The cognitive Maps of Political Elites / Ed. R. Axelrod. N.Y.: Princeton, 1976.
9. *Толмен Э.* Когнитивная карта у крыс и человека // Хрестоматия по истории психологии. М., 1980.
10. *Neisser U.* Cognitive psychology. N.Y., 1967.
11. *Девятко И.Ф.* Модели объяснения и логика социологического исследования.– М., 1996.
12. *Cicourel A.V.* Cognitive Sociology. L.: Pinguin Education, 1973.
13. *Loomis Ch., Dyer E D.* Social systems. Cambridge Mass, 1976.
14. Eden C. Cognitive mapping // Eur. J. of Operational Res. 1988. Vol. 36, № 1.
15. *Saeed K.* The dynamics of economic growth and political instability in developing countries // System Dynamics Review. 1986. Vol. 2, № 1.

Л.И. Замкова

РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ ЭФФЕКТИВНОЙ ОПЛАТЫ СЧЕТОВ

В настоящее время существует большое количество программных комплексов автоматизации различных задач бухгалтерского учета. Но задача оплаты счетов остается неавтоматизированной. В данной работе предлагается автоматизация задачи оплаты счетов-фактур.

1. Задача эффективной оплаты счетов и метод её решения

Задача заключается в следующем: существует множество неоплаченных счетов на текущий момент времени; для каждого счета известна сумма выплаты r'_i и крайний срок оплаты d_i; кроме того, задается текущий день, на который производится оплата d_f. Для каждого счета можно определить пеню $l_i=r_i*p_i*v_i$, которую необходимо выплатить при просрочке оплаты за v_i дней с момента d_i по d_f (процент пени p_i оговаривается в соответствующем пункте договора). Пусть D – часть средств, которую можно использовать для оплаты счетов. Из множества счетов необходимо выбрать те, которые можно оплатить в рамках суммы D, минимизируя суммарную пеню. Таких выборок может быть несколько. Решением задачи оплаты

счетов является выборка, максимизирующая остаток средств у предприятия при оплате счетов в рамках суммы D.

Эта практическая задача описывается двухкритериальной моделью о рюкзаке:

$$\sum_{i=1}^{n} l_i \cdot y_i \to \max$$

$$\sum_{i=1}^{n} r_i \cdot y_i \to \min$$

$$\sum_{i=1}^{n} r_i \cdot y_i \le D$$

$$y_i \in \{0,1\} \quad i = 1,2,\ldots,n$$

$$r_i = r'_i + l_i$$

Разработанный метод решения предполагает последовательную оптимизацию. Сначала определяется множество допустимых решений, удовлетворяющих первому критерию оптимизации, затем при удовлетворении первому критерию и ограничению выбирается решение, удовлетворяющее второму критерию. Таким образом, двухкритериальная постановка сводится к решению однокритериальной задаче о рюкзаке:

$$\sum_{i=1}^{n} l_i \cdot y_i \to \max$$

$$\sum_{i=1}^{n} r_i \cdot y_i \le D$$

$$y_i \in \{0,1\} \quad i = 1,2,\ldots,n$$

Постановка бухгалтерской задачи эффективной оплаты счетов опубликована в работе [1].

При разработке метода решения использовался способ формализации, представленный в публикации [2]. Метод включает два основных этапа: прямой проход и обратный. Цель прямого прохода – определить одно оптимальное решение. Однокритериальная задача о рюкзаке разбивается на n этапных задач:

$$\max \sum_{i=1}^{k} l_i \cdot y_i$$

$$\sum_{i=1}^{k} r_i \cdot y_i \le D \qquad .$$

$$k=1,\ldots n \quad y_i \in \{0,1\} \quad i=1,\ldots k$$

$$L(y) = \sum_{i=1}^{k} l_i \cdot y_i \qquad R(y) = \sum_{i=1}^{k} r_i \cdot y_i .$$

Для задачи размерности k строится множество допустимых решений ($C_{k,base}$). Элементы $C_{k,base}$ упорядочиваются по возрастанию значений целевой функции. $C_{1,base}$ содержит два вектора (0) и (1). $C_{k,base}$ k=2,3…n формируется на основе множества $C_{к-1,base}$. Очередной в порядке номеров вектор $\bar{c}_j=(c_1,c_2,\ldots,c_{k-1})$ j=1,2,...,$|C_{k-1,base}|$, принадлежащий $C_{k-1,base}$, порождает два вектора-потомка, размерности k $c^1=(c_1,c_2,\ldots,c_{k-1}, 0)$ и $c^2=(c_1,c_2,\ldots,c_{k-1},1)$. Если в $C_{k,base}$ нет векторов, дающих равное или большее

значение целевой функции, чем c^1, то вектор c^1 помещается на последнее место в $C_{k,base}$. Если $C_{k,base}$ есть вектор, доминирующий* над c^1, то c^1 заносится на последнее место во множество $C_{к,ots}$ – множество векторов, отсекаемых из $C_{k,base}$ на k-м шаге. Если в $C_{к,base}$ есть векторы, дающие большее значение целевой функции, то c^1 заносится перед первым таким вектором. После обработки первого вектора-потомка c^1 проверяется возможность добавления во множество $C_{k,base}$ второго потомка c^2. Если значение функции веса $R(c^2)$ не превосходит D, то вектор c^2 заносится на последнее место в $C_{k,base}$. Сформированное таким образом множество $C_{k,base}$ содержит вектора, упорядоченные по возрастанию значений целевой функции.

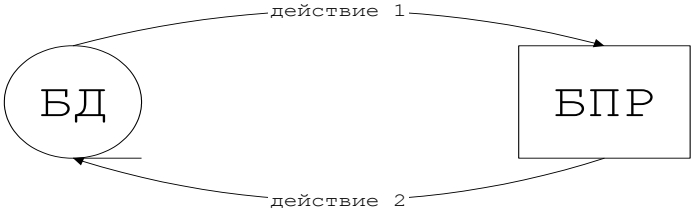
Выходными данными прямого прохода являются множества $C_{i,base}$, $C_{i,ots}$, i=1, 2…, n и вектор opt, который расположен в $C_{n,base}$ на последней позиции.

В результате функционирования алгоритма обратного прохода из всех решений, максимизирующих целевую функцию, выбирается наименьшее по значению функции веса. Введем множество B, элементами которого будут последние векторы множеств $C_{i,base}$ i=1, 2, …, n. Если последний вектор $C_{i,base}$ i=1, 2, …, n максимизирует целевую функцию, то он помещается в B. Вектор расширяется до размерности n нулевыми координатами. Он включается в C^{opt} (множество оптимальных решений) при выполнении условия несовпадения по составу**. Для каждого элемента b^k из множества B (k – размерность вектора b^k) определяются предки размерности k–1, k–2, …, 2. Переобозначим вектор b^k и его предков как вектора p^t t=k, k–1, …, 2 (t – их размерность). Если в $C_{t,ots}$ есть вектор, равный по значению целевой функции вектору p^t t=k, k–1, …, 2, то этот вектор расширяется до размерности k последними k–t координатами вектора b^k. Затем полученный вектор размерности k расширяется нулевыми координатами до размерности n. Расширенный таким образом вектор заносится в C^{opt} при выполнении условия несовпадения по составу с векторами, включенными в C^{opt}.

После завершения обработки всех векторов, принадлежащих множеству B, среди элементов C^{opt} выбирается наименьший по значению функции веса.

2. Описание программной системы, разработанной на основе метода

Программная система принятия решения об эффективной оплате счетов имеет следующую структуру:



Здесь БД – база данных «Бухгалтерия», БПР – блок принятия решения.

* Пусть y^1 и y^2 – два вектора размерности k. $L(y^1)$ и $L(y^2)$ – значения целевой функции в точках y^1 и y^2, $R(y^1)$ и $R(y^2)$ – значения функции веса в этих точках. Если $L(y^1) = L(y^2)$ и $R(y^1) \geq R(y^2)$, то вектор y^2 считается доминирующим над y^1.

** Два вектора являются совпадающими по составу, если они определяют одно подмножество множества $L=\{l_i\}$ i=1,2…n. Например, рассмотрим множество $L=\{l_1,l_2,l_3,l_4\}$, где $l_1=5$, $l_2=3$, $l_3=3$, $l_4=2$. Векторы (1101), (1011) совпадают по составу, так как они определяют подмножество {5,3,2}.

Исходными данными для БПР являются значения из таблицы «Сводная» (см. разд. 3) и таблицы «Сумма предприятия» базы данных «Бухгалтерия». Действие 1 исключает из таблицы «Сводная» записи, для которых значение поля «Признак» есть True. Таким образом, при очередном запуске системы рассматриваются только неоплаченные счета (значение поля «Признак» есть False). Перед запуском программной системы необходимо переформировать таблицу «Сводная», а при необходимости и таблицу «Сумма предприятия». Элементам массива l присваиваются значения множества значений поля «Пеня» таблицы «Сводная». Элементам массива r присваиваются значения множества значений поля «Сумма в счете» таблицы «Сводная». Переменной D присваивается значение поля «Sum_Sum_сумма» таблицы «Сумма предприятия». Блоком принятия решения определяется решение двухкритериальной задачи о рюкзаке:

$$\sum_{i=1}^{n} l_i \cdot y_i \rightarrow \max$$

$$\sum_{i=1}^{n} r_i \cdot y_i \rightarrow \min \qquad r_i = r_i' + l_i \;\; i = 1,2,\ldots,n,$$

$$\sum_{i=1}^{n} r_i \cdot y_i \le D$$

$$y_i \in \{0,1\} \quad i = 1,2,\ldots,n$$

где n – количество неоплаченных счетов на текущий момент, количество записей в таблице «Сводная» после действия 1.

Эта задача решается двухэтапным алгоритмом, описанным в разд. 1. Именно этот алгоритм заложен в основу программы, представляющей БПР. БПР реализован в Borland C++ Builder 4. БПР выдает вектор y, являющийся решением двухкритериальной задачи о рюкзаке. Вектор y имеет булевы координаты. Причем y_i=1 означает, что i-й счет выбран для оплаты, а y_i=0 означает, что i-й счет не будет оплачиваться. Таким образом, из исходного множества неоплаченных счетов выбирается подмножество счетов, которые можно оплатить на текущий момент. На базе решения y корректируется таблица «Сводная» – действие 2. Для счетов, определяемых y_i=1, устанавливается значение True поля «Признак» i-й записи. Таким образом, пользователь до запуска программной системы по базе данных «Бухгалтерия», а именно по таблице «Сводная» может просмотреть, какие счета-фактуры нужно оплатить.

При разработке программной системы особого внимания потребовал вопрос совместимости продукта фирмы Microsoft СУБД Access и продукта фирмы Borland C++ Builder (системы программирования). Это выражалось в том, чтобы приложение в Builder могло обратиться к таблицам базы данных «Бухгалтерия», созданной в СУБД Access. Для того чтобы это было возможным, необходимо с пакетом Builder устанавливать утилиту bdecfg32.exe. С помощью этой утилиты создается псевдоним базы данных «Бухгалтерия», с которым и работает приложение в Builder.

Для каждой таблицы, с которой будет работать приложение, создается экземпляр объекта типа TdataSource и экземпляр объекта типа TTable. Причем в свойстве DatabaseName компоненты Table указывается имя псевдонима. А в свойстве TableName – имя таблицы базы данных, для которой создается компонента. В свойстве DataSet компоненты DataSource указывается имя соответствующей таблицы базы данных. В приложении можно писать команды манипулирования с экземплярами объектов типа TTable, например, считывать и изменять значения полей.

На базе предприятия «Элегант» проводилось тестирование программной системы принятия решения об оплате счетов. В качестве исходных данных для программной системы использовались данные того же порядка, который фигурирует в реальной бухгалтерии предприятия «Элегант». Использовать конкретную информацию из бухгалтерских документов не предоставилось возможным из-за коммерческой тайны.

Рассмотрим тестирующий пример. Программная система была запущена 28.03.04. Пеня, сумма в счетах и средства предприятия выражены в рублях.

D⇔«Сумма предприятия»=35000

r'⇔«Сумма в счете1» ={320, 550, 1750, 1500,4000,1850,500,3700,300,250,180}.
«Процент пени»={0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2}
«период оплаты»={4,3,3,2,3,2,4,2,2,1,3}
«Дата выписки прих. ордера»={25.02.04, 20.03.04, 18.02.04, 01.03.04, 18.03.04, 20.02.04, 10.03.04, 12.03.04, 11.03.04, 15.02.04, 17.03.04}
Множество значений поля «Пеня» рассчитывает СУБД Access.

l ⇔«Пеня»={(28.03.04–(25.02.04+4)– 1)*0.2*320=1856,
(28.03.04–(20.03.04+3)–1)*0.2*550=660,
(28.03.04–(18.02.04+3)–1)*0.2*1750=12950,
(28.03.04–(01.03.04+2)–1)*0.2*1500=7800,
(28.03.04–(18.03.04+3)–1)*0.2*4000=6400,
(28.03.04–(20.02.04+2)–1)*0.2*1850=13320,
(28.03.04–(10.03.04+4)–1)*0.2*500=1500,
(28.03.04–(12.03.04+2)–1)*0.2*3700=11100,
(28.03.04–(11.03.04+2)–1)*0.2*300=960,
(28.03.04–(15.02.04+1)–1)*0.2*250=2100,
(28.03.04–(17.03.04+3)–1)*0.2*180=324}
Множество значений поля «Сумма в счете» рассчитывает СУБД Access.

r ⇔«Сумма в счете»={320+1856=2176,
550+660=1210,
1750+12950=14700,
1500+7800=9300,
4000+6400=10400,
1850+13320=15170,
500+1500=2000,
3700+11100=14800,
300+960=1260,
250+2100=2350,
180+324=504}

D=35000

i	1	2	3	4	5	6	7	8	9	10	11
l_i	1856	660	12950	7800	6400	13320	1500	11100	960	2100	324
r_i	2176	1210	14700	9300	10400	15170	2000	14800	1260	2350	504

В результате работы программной системы полю «Признак» таблицы «Сводная» присваиваются следующие значения:

«Признак»={true, false, true, false, false, true, false, false, false, true, true}

Это соответствует решению y=(1,0,1,0,0,1,0,0,0,1,1) двухкритериальной задачи о рюкзаке.

Вся пеня, которая идет на 28.03.04, составляет 58970 рублей. Из неё в рамках суммы предприятия 35000 рублей оплатили 30550 рублей, что составляет ≈52% от всей пени на 28.03.04. Кроме того, после оплаты выбранных счетов у предприятия остается 35000–34900=100 рублей при учете того, что из 35000 оплачиваются выбранные счета и пеня по ним.

Для этого примера программная система выдала результат за 1,1 минуты.

3. Проектирование базы данных «Бухгалтерия»

Реляционная база данных (БД) «Бухгалтерия» создавалась в системе управления базами данных (СУБД) Access97. БД состоит из восьми основных таблиц: «Выписки из банка», «Договоры», «Лист кассовой книги», «Счета», «Сводная», «Сумма в кассе», «Сумма на лицевом счете», «Сумма предприятия». Таблицы «Выписки из банка», «Договоры», «Лист кассовой книги», «Счета» заполняются на основании бухгалтерских документов. Представим подробно структуру этих таблиц и принцип их заполнения.

Структуру таблицы «Выписки из банка» составляют следующие поля: «Дата выписки», «Сумма на счете». Эта таблица заполняется на основании информации множества выписок из лицевого счета. Дата выдачи лицевого счета заносится в поле «Дата выписки», а исходящий остаток – в поле «Сумма на счете».

Структуру таблицы «Договоры» составляют следующие поля: «Наимен. документа», «Процент пени», «Период оплаты». В поле «Наимен. документа» заносится значение, например, **договор85** (номер договора), в поле «Процент пени» – данные раздела договора85 «Гарантии исполнения и ответственность сторон» (например, процент пени 0,15), в поле «Период оплаты» – данные раздела договора85 «Цены и порядок расчета» (например, срок оплаты 3 дня с момента получения приходного ордера).

Структуру таблицы «Счета» составляют следующие поля: «Наимен. договора», «Сумма в счете1», «Дата выписки прих. ордера», «Номер счета». В поле «Наимен. договора» заносится номер договора, по которому выставляется счет (договор указан как основание в накладной, прилагаемой к счету-фактуре, например, **договор85**). В поле «Сумма в счете» заносится сумма, выставленная в счете-фактуре (например, в счете-фактуре выставлена сумма 5087 руб. 25 коп). В поле «Дата выписки прих. ордера» заносится, например, дата 25.07.2001. В поле «Номер счета» заносится номер счета-фактуры, соответствующего указанному договору.

Структуру таблицы «Лист кассовой книги» составляют следующие поля: «Номер документа» и «Сумма». В поле «Номер документа» заносится номер приходного или расходного ордера, по которому фиксируется сумма в колонке приход или расход листа кассовой книги за текущий день. В поле «Сумма» значение из колонки приход заносится с положительным знаком, а колонки расход с отрицательным знаком. Остаток на начало дня из листа кассовой книги за текущий день заносится в поле «Сумма» с положительным знаком. Следует заметить, что таблица «Лист кассовой книги» заполняется по листу кассовой книги на текущий день.

Таблица «Сумма в кассе» формируется запросом на создание таблицы. Эта таблица содержит одну запись с единственным полем «Sum_сумма». Значение поля «Sum_сумма» формируется суммированием множества значений поля «Сумма» таблицы «Лист кассовой книги». Запросом на выборку по таблице «Выписки из банка» определяется запись, для которой поле «Дата выписки» имеет максимальное значение.

Таблица «Сумма на лицевом счете» формируется запросом на создание таблицы. Она содержит единственную запись, соответствующую ранее определенной максимальной дате (дате последней выписки из банка). Запросом на добавление к таблице «Сумма в кассе» добавляется значение поля «Сумма на счете» таблицы

«Сумма на лицевом счете». Таким образом, количество элементов множества значений поля «Sum_сумма» таблицы «Сумма в кассе» становится равным двум.

Таблица «Сумма предприятия» формируется запросом на создание таблицы. Она содержит единственную запись с одним полем «Sum_Sum_сумма», полученную суммированием значений поля «Sum_сумма» таблицы «Сумма в кассе».

Таблица «Сводная» формируется запросом на создание таблицы. Результаты запроса формируютя на основании двух связных таблиц, таблицы «Договоры» и таблицы «Счета». Таблицы связываются по полям «Наимен. документа» (таблица «Договоры») и «Наимен. договора» (таблица «Счета»). Таблицу «Сводная» составляют следующие поля: «Наимен. договора» (таблица «Счета»), «Процент пени» (таблица «Договоры»), «Период оплаты» (таблица «Договоры»), «Сумма в счете1» (таблица «Счета»), «Дата выписки прих. ордера» (таблица «Счета»), «Пеня», «Сумма в счете», «Признак».

Поле «Пеня» вычисляется согласно следующему выражению:

(Date()–(([дата выписки при ордера]+[период оплаты])–1))*
[процент пени][сумма в счете1]

Поле «Сумма в счете» вычисляется согласно выражению:

[сумма в счете1]+[пеня]

Поле «Признак» инициализируется значением False. Это означает, что ни один счет не выбран для оплаты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Замкова Л.И., Чефранов А.Г.* Алгоритм принятия решения об оплате счетов на основе решения задачи о ранце / Конф. «Новые информационные технологии. Разработка и аспекты применения».– Таганрог, 2000.
2. *Prudnicov S.P., Gorelova G.V.* Using the step Cut method // 14 Conf. on System modeling and optimization, 1989, Leipzig, DDR.

Е.Г. Иванова

УСЛОВИЯ ПОНИМАНИЯ ТЕКСТОВ НА ЕСТЕСТВЕННЫХ ЯЗЫКАХ

Введение

Исследование и моделирование реализуемого на уровне сознания в естественном интеллекте механизма *понимания* текстов на естественном языке является одной из наиболее важных задач в общей проблематике работ, проводимых с целью создания искусственного интеллекта. Решение этой задачи необходимо для обеспечения возможности *смыслового* обмена информацией между *искусственным* и *естественным* интеллектами на некотором упрощенном (для начала) подмножестве естественного языка. Поэтому вопросам, связанным с исследованиями тех или иных аспектов механизма *понимания* текстов на естественном языке, посвящено достаточно большое число работ [1, 2. С. 83–87, 210–215, 292–298].

В настоящей работе рассматривается связь между *допустимостью* текстов на естественном языке и *понимаемостью* этих текстов. Обосновывается утверждение, что допустимость (на всех уровнях) текстов на естественном языке при одинаковых исходных объемах базовых и оперативных знаний у источников и приемников информации является необходимым и достаточным условием для обеспечения возможности *понимания* этих текстов.