* * *

Приведенные результаты показывают, что для выработки управляющих воздействий, направленных на подъем промышленного производства, в условиях переходной экономики может быть использован изложенный выше макроэкономический подход. Для реализации этого подхода количественные показатели предварительно преобразуются в нечеткие переменные, которые затем подаются на вход логической нейронной сети, обучаемой на непредставительной выборке данных. Используемый алгоритм обучения позволяет минимизировать состав независимых переменных и архитектуру нейронной сети. Представление обученной нейронной сети в виде компактной системы логических функций позволяет сформировать решающую таблицу и выработать систему управляющих воздействий, направленных на стабилизацию производства в текущем месяце.

Дополнительная информация доступна в Интернете по адресу:

http://members.wbs.net/homepages/n/e/u/neurolab/.

Автор выражает признательность профессору S. Wermter (Sunderland University, UK) за полезное обсуждение результатов исследований.

Список литературы

1. **Римлер Ю.** Эконометрические методы анализа развития. М.: Статистика, 1979. 276 с.
2. **Бир С.** Мозг фирмы. М.: Радио и связь, 1993. 524 с.
3. **Леонтьев В.** Экономические эссе. М.: Политиздат, 1989. 467 с.
4. **Ивахненко А. Г.** Непрерывность и дискретность. Киев: Наукова Думка. 1990. 205 с.
5. **Parks P., Ivakhnenko A. et al.** A Self-Organization Model of the British Economy for Control with Optimal Prediction // Int. Journal Computational and Informational Science. 1975. N 4. P. 43—56.
6. **Ivakhnenko A. G., Ivakhnenko G. A., Müller J.-A.** Self-Organization of Nets of Active Neurons // Systems Analysis Modeling Simulation (SAMS). 1995. N 20. P. 93—106.
7. **Горбань А. Н., Россиев Д. А.** Нейронные сети на персональном компьютере. М.,1996. 276 с.
8. **Graven M., Shavlik J.** Learning Symbolic Rules Using Artificial Networks // Machine Learning: Proceedings of the Tenth Int. Conference. 1993. P. 38—41.
9. **Opitz D., Shavlik J.** Connectionist Theory Refinement: Genetically Searching the Space of Network Topologies // Artificial Intelligence Research. 1997. N 6. P. 177—209.
10. **Towell G., Shavlik J.** Extracting Refined Rules from Knowledge-Based Neural Networks // Machine Learning. 1993. Vol. 131. P. 71—101.
11. **Wermter S.** Knowledge Extraction from Transducer Neural Networks // Applied Intelligence. 1999. N 3. P. 1—18.
12. **Щетинин В. Г.** Анализ факторов экономического роста региона // Вопросы статистики. 1996. № 3. С. 40—46.
13. **Щетинин В. Г.** Анализ факторов гарантированного экономического роста региона // Регионология. 1996. № 1. С. 189—195.
14. **Щетинин В. Г., Костюнин А. В.** Принятие решений на нейронных сетях оптимальной сложности // Автоматизация и современные технологии. 1998. № 4. С. 38—43.
15. **Щетинин В. Г., Столярова О. В., Костюнин А. В.** Синтез решающих правил на нейронных сетях для управления производством // Приборы и системы управления. 1999. № 1. С. 72—77.
16. **Schetinin V. G., Kostunin A. W.** Self-Organization of Neuron Collective of Optimal Complexity // Proc. of Int. Symposium NOLTA'96. Japan, 1996. P. 245—248.
17. **Schetinin V. G.** Multi-layered Self-Organizing in Neural Networks of Optimal Complexity // Automatics and Computer Science. 1998. N 4. P. 30—37.

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ

УДК 002.53:004.451.52

С. А. Трофимов,
МГУПП

Некоторые подходы к поиску файлов данных в информационных системах

Обсуждаются возможные подходы к поиску файлов данных в информационных системах, основанные на эталонах адресов хранения файла на диске.

Информационные системы (ИС), определяющие основные параметры современной индустрии обработки информации, представляют собой сложный комплекс технических, программных, информационных и организационных средств, обеспечивающих централизованное накопление и коллективное использование информации. Информация хранится в файлах данных, с которыми взаимодействуют функциональные части ИС. Каждая ИС взаимодействует с некоторым множеством физических файлов данных $D = \{d_i\}$, $i = 1, Max$, где Max — общее число файлов данных, на работу с которым рассчитана конкретная ИС. Это число может значительно изменяться при использовании различных форматов хранения. Так, если программный интерфейс API позволяет работать с несколькими таблицами данных, физически представленными на диске как один файл, то $Max = 1$.

Для того чтобы получить дескриптор файла для последующей с ним работы, приложение в большинстве случаев должно передать операционной системе полный адрес хранения файла на диске. Небольшое приложение обычно сохраняет адреса хранения файлов или полностью список файлов данных вместе с адресами их хранения в файлах настройки или системном реестре. Сохранение происходит один раз во время установки приложения. В дальнейшем приложение использует следующую схему открытия файла:

- получить адрес доступа для конкретного файла;
- открыть файл;
- проверить, открыт ли файл;
- обработать возникшие ошибки;
- если файл открыт успешно, продолжить работу.

Обработка ошибок обычно одинакова для всех открываемых файлов и включает в себя аварийное завершение программы, если файл по каким-либо причинам открыть не удалось.

Достоинствами данного способа можно считать относительную простоту и распространенность применения, а также малое время поиска файла (есть по указанному адресу или нет).

К недостаткам можно отнести:

- малую гибкость доступа к файлам, например, невозможность переключаться между различными наборами файлов данных в течение одного сеанса работы (локальный/сетевой набор);
- невозможность обработки некоторых ошибок разделения доступа (например, разрешение чтения, но запрет для записи или удаления файлов по используемому адресу).

Основа этих проблем лежит в самом принципе поиска файлов. В данном случае адрес доступа является свойством файла данных, таким как его имя или расширение:

$$S_i = \{s_1, s_2, s_3, s_4, ..., s_x\},$$

где, например, s_1 — адрес хранения; s_2, s_3 — имя и расширение; s_4, ..., s_x — другие свойства, не рассматриваемые в данном случае.

Для небольшого приложения такой подход вполне оправдан. Но для информационной системы, в которой возможна работа с сотнями файлов, находящихся на различных сетевых компьютерах как в локальной, так и в глобальной сетях, такой подход ограничит функциональные возможности и может снизить общую устойчивость системы.

Изменим принцип поиска файлов в ИС. Сделаем основной единицей хранения не файл, а адрес. Получим некоторое множество адресов хранения C, у каждого j-го элемента которого имеется некоторый набор свойств A_j:

$$C = \{c_j\},$$
$$A_j = \{a_1, a_2, a_3, a_4, a_5, ..., a_n\},$$

где, например, a_1 — имя; a_2 — атрибуты; a_3 — размер доступного пространства; a_4 — множество хранящихся по адресу файлов; a_5, ..., a_n — другие, не рассматриваемые здесь, свойства. Причем ИС хранит *требуемые* атрибуты каталога, такие как доступность на чтение/запись/создание/удаление файлов и необходимый на сеанс работы размер дискового пространства, а также, возможно, и другие необходимые для конкретной ИС свойства. При запуске модуля ИС, в котором проводится работа с определенными файлами, проверяются адреса хранения на соответствие *реальных* атрибутов требуемым и заполняется множество реально находящихся по адресу файлов, которые впоследствии сравниваются с требуемым списком файлов.

Допустим, что модуль ИС имеет некоторое множество функций P, каждый l-й элемент функции работает с множеством файлов F_l и множеством эталонных адресов CP_l:

$$P = \{p_l\};$$
$$F_l = \{f_k\},\ CP_l = \{cp_r\}.$$

Определим множество свойств f_k как Sf_k:

$$Sf_k = \{s_2, s_3, s_4, s_5, ..., s_x\}.$$

Следует заметить, что f_k в отличие от d_i не имеет свойства s_1 — пути доступа, однако имеется свойство s_4, назовем его алгоритмической группой. Данный признак объединяет файлы, у которых общее одно из следующих свойств: физическое расположение, совершаемые над ними физические операции, такие как чтение/запись/создание/удаление или другие свойства. В алгоритмические группы входят наборы файлов данных, т. е. группы файлов, которые нельзя разделять без нарушения целостности данных. Исходя из наличия алгоритмических групп и дополнительных требований внутренних алгоритмов выводится множество адресов, с которыми будет работать каждая функция p_l, т. е. эталонное множество CP_l.

При запуске модуля ему передается список возможных адресов хранения C, зарегистрированных в системе. Модуль проверяет полученные адреса и определяет множество адресов CT_l, где $CT_l = C \cap CP_l$, причем каждый элемент CT_l должен существовать физически.

При таком подходе система приобретает значительную гибкость. Модуль информационной системы может:

- отключать отдельные режимы, для которых $CT_l = \varnothing$ или ни в одном из элементов CT_l нет необходимого набора файлов данных, и оставлять работоспособными остальные режимы, для которых эти условия выполняются;
- в зависимости от алгоритма переключаться между различными наборами файлов данных, ес-

ли число элементов CT_l, в которых присутствуют одинаковые наборы, больше 1;

- до открытия файлов определять комплектность наборов файлов с возможностью выдачи запроса пользователю для принятия решения или с автоматическим восстановлением недостающих файлов;
- автоматически переключаться на запасной набор данных при недоступности основного адреса;
- автоматически осуществлять синхронизацию данных в различных адресах хранения.

Рассмотрим применение такого подхода на примере.

Допустим, что имеется рабочая станция, на которой установлена некая информационная система, подключенная к локальной сети $LS1$ посредством сетевого адаптера и к локальной сети $LS2$ посредством модема. Один из модулей этой информационной системы требует для своей работы шесть информационных единиц, представленных табл. 1, где каждая информационная единица F может состоять как из отдельного файла, так и включать в себя набор данных.

Таким образом, имеется три алгоритмические группы файлов. Исходя из этого получаем эталонный набор адресов, представленный в табл. 2.

В табл. 2 R — доступность для чтения, W — доступность для записи, C — доступность для создания, D — доступность для удаления. Плюсами в колонке "доступ" обозначены обязательные условия, минусами — условия, которые могут присутствовать, но не обязательны для работы. Плюс в колонке "расположение" указывает, какой адрес взять при прочих равных условиях.

Допустим, что в информационной системе зарегистрированы имена адресов доступа C1, C2, C3, C4. Рабочая станция, на которой установлена информационная система, соединена с помощью локальной сети с файловым сервером FS1 и с помощью удаленного доступа — с файловым сервером FS2. Табл. 3 показывает расположение адресов и наличие по ним файлов перед запуском модуля.

Таблица 1

Файлы	Назначение	Требуемый доступ	Расположение
F1, F2	Справочники	Чтение	
F3, F4	Рабочие данные	Чтение/запись	
F5, F6	Временные данные	Чтение/запись/создание/удаление	На наиболее быстром дисковом устройстве

Таблица 2

Адрес	Назначение	Доступ				Расположение	
		R	W	C	D	станция	сервер
CP1	Справочники	+	–	–	–	+	
CP2	Рабочие данные	+	+	–	–		+
CP3	Временные данные	+	+	+	+	+	

Таблица 3

Адрес	Файлы	Доступ				Расположение	
		R	W	C	D	станция	сервер
C1		+	+	+	+	+	
C2	F1, F2, $F3_1$, $F4_1$	+					FS1
C3	$F3_2$, $F4_2$	+	+				FS1
C4	$F3_3$, $F4_3$	+	+				FS2

Согласно данной таблице C1 — локальный адрес без ограничений по доступу, C2 — сетевой адрес, доступный только для чтения, C3 — сетевой адрес, доступный для чтения/записи на файловом сервере 1, C4 — сетевой адрес, доступный на файловом сервере 2. На C2, C3, C4 имеются различные версии рабочих файлов F3, F4, причем на C2 — только структуры для создания дубликатов.

Допустим, что при первом запуске адрес C4 недоступен. Происходит исследование зарегистрированных адресов доступа на соответствие эталонным. Получаем таблицу соответствий (табл. 4).

При подключении к FS2 получаем табл. 5.

Заметим, что C2 не был использован для рабочих данных по ограничениям эталона CP2 несмотря на то, что там имеются версии файлов $F3_1$, $F4_1$, а адрес C1 был использован даже без находящихся по нему файлов.

Система прекрасно работает как при отключенном сервере FS2, так и при подключенном. Система автоматически или по запросу пользователя может переключаться между различными наборами данных C3, C4 или выполнять их зеркальное копирование.

Таблица 4

Эталонный адрес	Доступный адрес
CP1	C2
CP2	C3
CP3	C1

Таблица 5

Эталонный адрес	Доступный адрес
CP1	C2
CP2	C3, C4
CP3	C1

Таблица 6

Адрес	Файлы	Доступ				Расположение	
		R	W	C	D	станция	сервер
C1	F1, F2, $F3_2$, $F4_2$	+	+	+	+	+	
C2	F1, F2, $F3_1$, $F4_1$	+					FS1
C3	$F3_2$, $F4_2$	+	+				FS1
C4	$F3_3$, $F4_3$	+	+				FS2

Таблица 7

Эталонный адрес	Доступный адрес
CP1	C1
CP2	C3, C4
CP3	C1

Рассмотрим случай, когда на рабочей станции создается локальная копия файлов F1, F2, $F3_2$, $F4_2$, как представлено в следующей табл. 6.

При таком расположении файлов получаем увеличение быстродействия системы, так как таблица соответствия изменится (табл. 7).

Теперь для работы системы используется локальная версия неизменяемых справочников, находящихся на C1, причем для такого переключения нет необходимости что-либо изменять в настройках информационной системы. В таком варианте нестрашно полное отключение станции от локальной сети. Система выдаст предупреждение пользователю и перейдет к автономной работе, а при последующем восстановлении сети также без каких-либо изменений настроек перейдет к сетевой работе.

Эти примеры демонстрируют лишь немногие из имеющихся возможностей поиска файлов, основанного на эталонах адресов хранения. Но они показывают, что такой подход оправдан при:

- достаточно частом изменении конфигурации системы, когда используются сменные носители информации или удаленный доступ к сети;
- необходимости одновременной работы с несколькими наборами данных;
- достаточном числе пользователей системы с разными категориями доступа к каталогам.

* * *

В целом, данный подход значительно уменьшает трудозатраты по администрированию системы, повышает гибкость системы, уменьшает число запросов к дискам при поиске файлов, так как осуществляется поиск не отдельных файлов, а алгоритмической группы файлов. Таким образом, при выборе метода поиска файлов, в конечном итоге, необходимо исходить из требований системы и разумного баланса между удобством администрирования и временем поиска файлов.

Список литературы

1. **Ломако Е. И.** Макетирование, проектирование и реализация диалоговых информационных систем. М.: Финансы и статистика, 1993. 320 с.
2. **Липаев В. В., Филинов Е. Н.** Мобильность программ и данных в открытых информационных системах. М.: Научная книга, 1997. 368 с.
3. **Microsoft** Corporation. Руководство программиста по Microsoft Windows 95 / Пер. с англ. М.: Издательский отдел "Русская Редакция" ТОО "Channel Trading Ltd.", 1997. 600 с.

Указатель статей, опубликованных в журнале "Информационные технологии" в 1999 г.

ОБЩИЕ ВОПРОСЫ ИНФОРМАТИЗАЦИИ

Волгин Л. И. Континуальные логики и предметные алгебры, порождаемые функцией взвешенных степенных средних. № 9.

Липаев В. В. Сертификация систем качества предприятий, разрабатывающих программные средства для информационных систем, на соответствие стандартам серии ISO 9000. № 12.

Штрик А. А. Обзор основных принципов государственной политики в области информационных технологий в развитых странах. № 10.

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Зверев Г. Н. Оценка точности логических приближений и границ применимости классической и неклассических логик в системах моделирования и принятия решений. № 12.

Мелешко В. Н. Математическое моделирование процесса сбалансированного развития сложных систем. № 10.

Севастьянов П. В., Вальковский В. И. Методика нечетко-интервального имитационного моделирования технико-экономических систем. № 6.

Таничев И. Н. Численно-аналитические алгоритмы вычисления функций матрицы и их приложения. № 6.

МЕТОДЫ ОПТИМИЗАЦИИ

Антонова Г. М. Моделирование процессов для поиска рационального решения. № 11.

Левин В. И. Интервальный подход к оптимизации в условиях неопределенности. № 1.

Левин В. И. Задачи непрерывной оптимизации в условиях интервальной неопределенности. № 7.