УДК 683.321

# ВИКОРИСТАННЯ МЕТОДУ ІНДУКТИВНОГО ВИВЕДЕННЯ ДЛЯ ВДОСКОНАЛЕННЯ ОНТОЛОГІЇ ПРЕДМЕТНОЇ ОБЛАСТІ ПОШУКУ

**Ю.В. РОГУШИНА, І.Ю. ГРИШАНОВА**

Пропонується для підвищення релевантності пошуку інформації використовувати знання предметної області пошуку, подані як онтології в інтероперабельній формі, придатній для автоматизованого оброблення (приміром, інформаційно-пошуковими агентами). Розроблено алгоритм автоматизованого доповнення онтологій новими термінами шляхом індуктивного узагальнення властивостей, обраних користувачем повнотекстових документів.

## ВСТУП

На сучасному етапі розвитку інформаційних технологій (ІТ) основні їх напрямки пов'язані зі створенням інформаційних систем (ІС), що використовують та обробляють знання відповідних предметних областей (ПрО). Тому зараз велике значення надається засобам подання знань в ІС, питанням їх інтероперабельності, розподіленого та повторного використання.

Знання є результатом пізнання дійсності, відображеної у свідомості людини у вигляді уявлень, понять, суджень і теорій. Знання ІС щодо ПрО — це сукупність відомостей ПрО (фактів, правил, властивостей об'єктів), які зберігаються у базі знань (БЗ) інтелектуальної ІС. Знання ПрО поділяються на факти, що відносяться до певної ПрО, закономірності, характерні для цієї ПрО, та гіпотези про можливі зв'язки між явищами, процесами і фактами, а також процедури для вирішення типових задач у даній ПрО. ІС, які базуються на знаннях, забезпечують більш повне та ефективне задоволення інформаційних потреб користувачів (потреб, що виникають, коли ціль користувача у процесі його професійної діяльності або в його соціально-побутовій практиці не може бути досягнута без залучення додаткової інформації з певних документів, доступ до яких має ІС).

Пошук інформаційних ресурсів (ІР) є важливою складовою таких актуальних напрямків розвитку ІТ, як електронна комерція, електронний бізнес, дистанційна освіта, телемедицина тощо. Використання знань ПрО, представлених у різних формах (текст, графіка, мультимедіа тощо), підвищує ефективність пошуку в Інтернеті, локальній мережі, на окремому комп'ютері. Якщо ІС має відомості щодо того, до якої ПрО відносяться інформаційні потреби користувача, то вона має можливість конкретизувати пошуковий запит, що відображає інформаційну потребу користувача. Так, відповідно до знань ПрО, потреба користувача у діагностиці може інтерпретуватися як запит щодо консультації лікаря або перевірки функціонування комп'ютера.

Створення БЗ ПрО — складний процес, який потребує багато часу та коштів. Тому доцільно здобуті знання ПрО використовувати багаторазово у

різноманітних задачах, що постають перед користувачами у цій ПрО. Наприклад, БЗ, створену для навчання у певній галузі, можна застосовувати для пошуку ІР у цій галузі, е-бізнесу при створенні експертних та консультуючих систем тощо. Для повторного використання знань ПрО необхідно подавати їх в інтероперабельній формі, придатній для автоматизованого оброблення, приміром, на основі онтологій [1].

## ПОСТАНОВКА ЗАДАЧІ

Пропонується для автоматизації створення БЗ у різних ПрО використовувати метод індуктивного виведення для узагальнення відомостей, що містяться в ІР (як повнотекстових, так і мультимедійних), релевантних ПрО. Щоб такі БЗ були інтероперабельними та придатними для повторного використання, доцільно подавати здобуті знання через онтології та тезауруси.

## ОНТОЛОГІЧНЕ ПОДАННЯ ЗНАНЬ

Термін *онтологія* прийшов з філософії. Онтологія — це угода про спільне використання понять, що містить засоби подання предметних знань і домовленості про методи міркувань. Неформально онтологія може розглядатися як певний опис погляду на світ у конкретній сфері інтересів, що складається з набору термінів і правил використання цих термінів, які обмежують їх значення в рамках конкретної ПрО. Онтології дозволяють подавати знання у формі, придатній для машинної обробки.

Онтологія — це БЗ спеціального виду, яка містить семантичну інформацію з певної ПрО. Компоненти онтології залежать від парадигми подання, але практично всі моделі онтології містять певні *концепти* (поняття, класи), *властивості* концептів (атрибути, ролі), *відношення* між концептами (залежності, функції) та додаткові *обмеження,* що визначаються аксіомами. Концептом може бути опис задачі, функції, дії, стратегії, процесу міркування тощо [2].

На формальному рівні онтологія — це система з наборів понять і тверджень про ці поняття, на основі яких можна будувати класи, об'єкти, відношення, функції та теорії. Формальна модель онтології $O$: впорядкована трійка $O = \langle T, R, F \rangle$, де $T$ — скінченна множина термінів ПрО, яку описує $O$; $R$ — скінченна множина відношень між термінами заданої ПрО; $F$ — скінченна множина функцій інтерпретації, заданих на термінах і/або відношеннях $O$. Відношення представляють тип взаємодії між концептами ПрО. Приклад бінарного відношення — «є частиною». Аксіоми використовуються для моделювання тверджень завжди істинних.

Моделі онтології класифікуються таким чином:

- прості моделі (мають лише концепти);
- на основі фреймів (лише концепти і властивості);
- на основі логік (Ontolingua, DAML+OIL).

Значна частина знань, набутих людством, доступна сьогодні через мережу Інтернет. Багато зусиль (приміром, проект Semantic Web консорціуму

W3C [3]) спрямовано на перетворення Інтернет на розподілене сховище знань. Один із перспективних напрямків такої роботи пов'язаний з онтологічним поданням знань про IP та Web-сервіси.

Поширення онтологічного підходу до подання знань привело до створення різноманітних мов подання онтології та інструментальних засобів, призначених для їх редагування та аналізу.

**OWL**. Онтологія OWL (Web Ontology Language) є послідовністю аксіом і фактів, а також посилань на інші онтології. Крім того, вона містить компоненти для запису авторства та іншої подібної інформації. OWL є документом Web. На неї можна посилатися через URI (Universal Resource Identificator). Мова подання OWL базується на DAML+OIL. OWL дозволяє усунути деякі обмеження порівняно з DAML+OIL, а також здатна прямо вказувати симетричність властивості.

Найбільш фундаментальні поняття ПрО мають відповідати класам, які знаходяться в корені різних таксономічних дерев. Кожен екземпляр в OWL належить до класу `owl:Thing`, а кожен визначений користувачем клас автоматично є підкласом `owl:Thing`. Специфічні для ПрО кореневі класи визначаються простим оголошенням іменованого класу. OWL також визначає порожній клас `owl:Nothing`. Визначення можуть бути розширюваними і розподіленими.

Фундаментальним таксономічним конструктором для класів є `rdfs:subClassOf`. Він зв'язує більш окремий клас з більш загальним. Якщо $X$ — підклас $Y$, то кожен представник $X$ також представник $Y$. Відношення `rdfs:subClassOf` є транзитивним. Якщо $X$ — підклас $Y$ і $Y$ — підклас $Z$, то $X$ — підклас $Z$.

Визначення класу складається з двох частин: вказуються назва або посилання і список обмежень. Кожний вираз, який безпосередньо міститься у визначенні класу, уточнює властивості представників цього класу, що належать до перетину зазначених обмежень.

Для визначення екземпляра досить оголосити його членом якогось класу.

Властивості дозволяють затверджувати загальні факти про члени класів і про екземпляри. Властивість — це бінарне відношення. Розрізняють два типи властивостей:

1) *властивості-значення* — відношення між представниками класів і RDF-літералами або типами даних, обумовлених XML Schema;

2) *властивості-об'єкти* — відношення між представниками двох класів.

При визначенні властивості існує багато способів обмежити ці відношення. Можна визначити домен і діапазон, спеціалізацію (підвластивість) існуючої властивості. Можливі й більш складні обмеження. Властивості, так само як класи, можуть бути організовані в ієрархію.

OWL використовує більшість вбудованих типів XML Schema. Посилання на ці типи здійснюються за допомогою URI для типів `http://www.w3.org/2001/XMLSchema`.

Формальна семантика OWL описує, як отримати логічні наслідки, маючи таку онтологію, тобто здобути факти, що не представлені в онтології

безпосередньо, проте випливають з її семантики. Ці наслідки можуть бути засновані на одному документі або множині розподілених документів, які комбінуються з використанням спеціальних механізмів OWL. OWL відрізняється від схеми XML тим, що це подання знання, а не формат повідомлень.

Однією з переваг OWL є наявність інструментального ПЗ, призначеного для аналізу знань, поданих мовою OWL. Ці інструменти забезпечують загальну підтримку онтологічного аналізу, яка не є специфічною для певної ПрО.

OWL має три діалекти, що різняться за виразністю: OWL Lite, OWL DL та OWL Full.

OWL Lite — найпростіший варіант, для користувачів, яким потрібна класифікація ієрархії. Вони використовують прості обмеження. OWL Lite забезпечує швидку міграцію тезаурусів та інших таксономій.

OWL DL орієнтований на користувачів, яким потрібна максимальна виразність без утрати повноти обчислень та гарантованого завершення всіх обчислень у визначений час. OWL DL містить всі язикові конструкції OWL з обмеженнями поділу типу (клас не може бути приватною властивістю, а властивість не може бути індивідом або класом). Назва OWL DL пов'язана з його відповідністю дескриптивній логіці.

OWL Full призначається для користувачів, яким потрібна максимальна виразність і синтаксична потужність RDF без обчислювальних гарантій. Наприклад, у OWL Full клас може одночасно розглядатися і як сукупність екземплярів, і як екземпляр. Інша істотна відмінність від OWL DL у тому, що `owl:DatatypeProperty` може бути позначене як `owl:InverseFunctionalProperty`. OWL Full припускає онтології, що розширюють склад визначеного словника RDF або OWL.

OWL використовує механізми RDF для типів даних. OWL Full може розглядатися як розширення RDF, а OWL Lite і OWL DL — як розширення обмеженого варіанта RDF. Кожен документ OWL (Lite, DL, Full) — це документ RDF, і кожен документ RDF — документ OWL Full, але тільки деякі RDF-документи можуть бути припустимими документами OWL Lite чи OWL DL. Через це потрібно дотримуватися деякої обережності у випадку здійснення користувачем міграції документів RDF у OWL. Коли виразність OWL DL або OWL Lite вважається прийнятною, проводяться деякі запобіжні заходи, щоб упевнитися, чи відповідає оригінальний RDF-документ додатковим обмеженням, які накладаються OWL DL і OWL Lite. Зокрема, треба упевнитися: тип кожного URI, що використовується як назва класу, визначений як owl:Class; кожен екземпляр належить, принаймні, одному класу (навіть якщо це тільки owl:Thing); URI для класів, властивостей та індивідуумів не перетинаються.

OWL DL і OWL Full використовують той самий словник, хоча OWL DL має деякі обмеження: вимагає поділу типу (клас не може також бути екземпляром або властивістю, а властивість не може бути також екземпляром або класом). Крім того, OWL DL вимагає, щоб властивості були визначені або як властивості–об'єкти, або як властивості–значення. Властивості–значення — це відношення між представниками класів, літералами RDF і типами да-

них XML Schema, а властивості – об'єкти — це відношення між представниками двох класів.

OWL припускає відкритість, тобто клас спочатку може бути визначений в одній онтології, а потім — розширений в інших онтологіях. Внаслідок цього судження є монотонними: нова інформація не спростовує попередню, факти і наслідки тільки додаються до онтології і не видаляються. Тому інформація може бути суперечною, і розробник онтології повинен це враховувати.

## ОНТОЛОГІЧНИЙ АНАЛІЗ ЗНАНЬ

Методологічною основою обробки знань ПрО є *онтологічний аналіз*. Рівень аналізу знань, який отримав назву онтологічного, запропоновано у роботі [5]. В основі цього підходу лежить опис системи в термінах сутностей, відношень між ними і перетворення сутностей, що виконується у процесі рішення певної задачі.

Для структурування знань ПрО використовують три основні категорії:

1) *статичну онтологію,* до якої входять сутності ПрО, їхні властивості і відношення;

2) *динамічну онтологію,* що визначає ситуації, які виникають у процесі рішення проблеми, і спосіб перетворення одних ситуацій в інші;

3) *епістемічну онтологію,* що описує знання, які керують процесом переходу від однієї ситуації до іншої.

У цій схемі є певна відповідність до рівнів концептуалізації знань і епістемічного аналізу в структурі. Розглянута схема онтологічного аналізу досить абстрактна, але її цінність у спрощенні аналізу слабо структурованих задач. Онтологічний аналіз припускає, що проблема, яка розв'язується, може бути зведена до проблеми пошуку, але при цьому не розглядається, яким саме способом потрібно виконувати пошук.

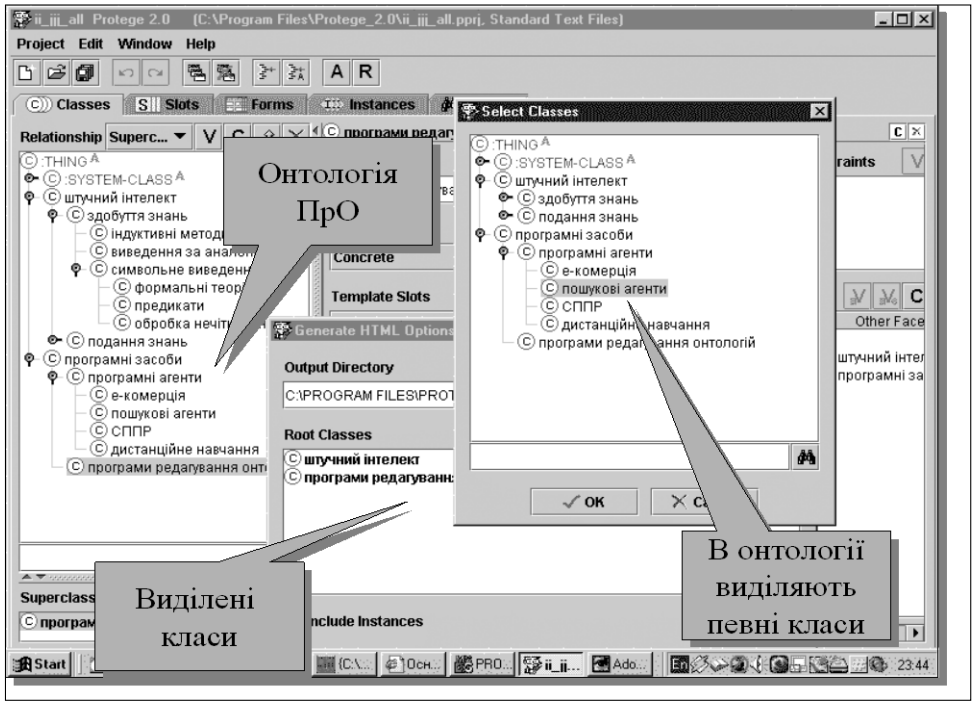
На сьогодні існують різноманітні методики створення онтологій ПрО (одна з найбільш поширених — IDEF5), проте всі вони досить складні і вимагають значних зусиль як з боку експертів ПрО, так і з боку інженерів зі знань. Слід відмітити, що зараз розроблено та вільно поширюється багато інструментальних засобів для створення, редагування, аналізу та візуалізації онтологій. Це значною мірою полегшує створення онтологій ПрО.

## ІНСТРУМЕНТАЛЬНІ ЗАСОБИ СТВОРЕННЯ ОНТОЛОГІЙ

*Protégé* [6] — локальна, вільно розповсюджувана Java-програма, що призначена для побудови (створення, редагування і перегляду) онтологій ПрО. Вона містить редактор онтологій, який дозволяє проектувати онтології, розгортаючи ієрархічну структуру абстрактних та конкретних класів і слотів. На основі сформованої онтології Protégé дозволяє генерувати форми для вв едення екземплярів класів і підкласів.

Protégé використовує фреймову модель подання знання OKBC (Open Knowledge Base Connectivity), яка дозволяє адаптувати його для редагування

моделей ПрО, поданих не в OWL, а в інших форматах ( UML, XML, SHOE, DAML+OIL, RDF і RDFS тощо). Цікава особливість Protégé користувач може виділити певні класи онтології (а також, якщо потрібно, екземпляри цих класів) та зберегти цю інформацію у форматі html (див. рисунок).




Фрагмент подання онтології ПрО «Штучний інтелект» в Protégé

Якщо користувач має досить сталу область інформаційних потреб (приміром, він займається науковими дослідженнями у цій галузі або виробляє продукцію певного типу), тоді доцільно для підвищення ефективності його роботи створити формальну модель цієї ПрО у вигляді онтології та використовувати її потім у якості БЗ для потрібних користувачеві інтелектуальних ІС. Приміром, така БЗ може бути корисною для персоніфікації інформаційного пошуку, дозволяючи задавати контекст пошукових запитів, у системах е-комерції (для вибору найбільш придатного товару) та дистанційному навчанні (для вибору найбільш придатного курсу) тощо.

Користувач створює онтологію тієї області, до якої відносяться його інформаційні інтереси, і потім використовує її при пошуку найбільш придатних ресурсів. Це досить складна задача. Звичайно, він може використовувати якусь загальну онтологію, що була створена раніше іншими дослідниками й покриває область його інтересів. Але через складність структури та великий обсяг таких онтологій користувачеві важко вносити до них зміни й доповнення. Крім того, загальні онтології можуть не відповідати переконанням і знанням користувача та не відображати його персональні переваги. З іншого боку, для самостійного створення онтологій користувач має не тільки чітко уявляти структуру ПрО, основні її поняття і зв'язки між ними, але й володіти знаннями та навичками інженера зі знань. Тому крім редактора онтологій, який дозволяє користувачеві безпосередньо

будувати онтологію, доцільно надати йому певні програмні засоби, що допоможуть формалізації знань.

## ІНДУКТИВНЕ УЗАГАЛЬНЕННЯ — МЕТОД ЗДОБУТТЯ ЗНАНЬ ПрО

Однією з проблем розробника онтології є необхідність виявити усі терміни, які характеризують об'єкти ПрО, та правильно встановити відношення між ними. Для її вирішення доцільно застосовувати методи індуктивного узагальнення, що дозволяють здобути терміни та зв'язки між ними з документів ПрО. У цьому випадку знання користувача щодо ПрО відображаються через те, які ІР він вважає релевантними цій ПрО (це можуть бути, наприклад, знайдені користувачем в Інтернеті документи, власні звіти тощо).

Користувач має сформувати $I$ — множину ІР, які відносяться до обраної ПрО. Припустимо, користувач хоче ввести до онтології нові поняття: $A$ та його підкласи (екземпляри, властивості тощо) $B_i$, $i=\overline{1,n}$, яким в OWL відповідає таксономічний конструктор класів rdfs:subClassOf.

У множині $I$ користувач виділяє підмножину з $m$ елементів: $I(A): I(A) \subseteq I$. $IP_j \in I(A)$, $j=\overline{1,m}$ — ресурси, пов'язані з поняттям $A$. Потім у цій множині $I(A)$ користувач виділяє підмножини $I(B_1),...,I(B_n)$, $i=\overline{1,n}$, які характеризують $B_i$, $i=\overline{1,n}$ — підкласи $A$.

На ці множини накладаються певні обмеження:

- $I(B_i) \subset I(A)$, $i=\overline{1,n}$.
- $\forall i, j$, $i=\overline{1,n}$, $j=\overline{1,n}$, $I(B_i) \cap I(B_j) = \varnothing$.
- $\bigcup_{i=1}^{n} I(B_i) = I(A)$, $i=\overline{1,n}$.

Після цього для кожного документа $IP_j \in I(A)$, $j=\overline{1,m}$ з множини $A$ будується словник $\mathrm{Ont}(IP_j) = \{t_k\}$, $k=\overline{1,p}: t_k \in T$, $t_k \in IP_j \in I(A)$ — множина термінів онтології користувача $O$, що містяться у певному документі та його метаописі. $\mathrm{Ont}(I(A))$ створюється як перетин множин онтологічних термінів ІР, що входять у відповідну множину $\mathrm{Ont}(I(A)) = \bigcap_{j=1}^{m} \mathrm{Ont}(IP_j)$. Аналогічно створюються $\mathrm{Ont}(I(B_i))$, $i=\overline{1,n}$.

Описом термінів $B_1,...,B_n$ можуть бути множини онтологічних термінів $\mathrm{Term}(B_i, A)$, $i=\overline{1,n}$, які отримують з множин $\mathrm{Ont}(I(B_i))$, $i=\overline{1,n}$ видаленням термінів з $\mathrm{Ont}(I(A))$, що усередині множини $A$ стають стоп-словами.

Множини $\mathrm{Term}(B_i, A)$, $i=\overline{1,n}$ можуть перетинатися (приміром, термін онтології $t_1$ міститься у множинах $\mathrm{Term}(B_1, A)$ та $\mathrm{Term}(B_2, A)$), тоді як нам

потрібно знайти саме ті терміни, за допомогою яких $B_i$ відрізняється від усіх інших.

Використаємо для цього метод індуктивного узагальнення ID3m, який розширює відомий алгоритм ID3 на випадок довільної кількості класів і враховує ступінь доступності інформації про значення різноманітних атрибутів [10].

Приклади навчальної вибірки — це ІР з $I(B_1),..., I(B_n)$, $i=\overline{1,n}$. Стовпцям навчальної вибірки відповідають онтологічні терміни з множини $T=\bigcup_{i=1}^{n}\text{Term}(B_i, A)$. Значення комірок — кількість входжень терміну до ІР. Параметром, за яким здійснюється класифікація, є належність ІР до певної множини з $I(B_1),..., I(B_n)$, $i=\overline{1,n}$.

Через те що алгоритм ID3m призначений для обробки не кількісних, а якісних даних, значення комірок перетворюють у якісні оцінки (як правило, використовують три значення: «відсутній», «мало» та «багато»).

На кожному кроці роботи алгоритму обчислюється, який онтологічний термін несе найбільше інформації, потрібної для класифікації ІР, за формулою

$$C(A_m)=\sum_i\sum_j \frac{C(A_m=a_{mi},\ R=R_j)}{T(A_m)}=$$

$$=\max_s C(A_s)=\max_s \sum_i\sum_j \frac{C(A_s=a_{si}, R=R_j)}{T(A_m)},$$

де $C(X,Y)$ — кількість інформації $C(X,Y)=\sum_i\sum_j p(X=x,\ Y=y)*$ $*\log p(X=x, Y=y)$; $p(X=x, Y=y)$ — можливість спільного наступу подій $X=x$ та $Y=y$; $T(A_m)$ — вартість одержання значення $A_m$.

Результатом роботи алгоритму є класифікаційне дерево, листи якого множини $I(B_1),..., I(B_n)$, $i=\overline{1,n}$; вузли — терміни онтології користувача $O$, а гілки — наявність цих термінів у документах. Застосовуючи таке дерево, користувачеві простіше розширити свою онтологію термінами $A$ та $B_i$, $i=\overline{1,n}$, вживаючи для їх визначень онтологічні терміни, що несуть найбільше інформації, та встановлюючи коректні відношення між новими та старими термінами онтології, які увійшли до класифікаційного дерева.

## ВИСНОВКИ

Використання алгоритму індуктивного узагальнення для обробки документів, які характеризують інформаційні потреби користувача та відповідають його уявленню щодо певних понять ПрО, дозволяє спростити процедуру розширення онтології та встановити коректні відношення між новими й старими поняттями ПрО.

**ЛІТЕРАТУРА**

1. *Артемьева Л.А., Гаврилова Т.Л., Клещев А.С.* Логические модели второго порядка для предметных областей // Информационный анализ. — 1997. — Вып. 6. — С. 14–21.
2. *Asuncion G.* Ontological Engineering: with Examples from Areas of Knowledge Management, e-Commerce and the Semantic Web (Advanced Information and Knowledge Processing). — http://www.amazon.com/gp/reader/1852335513/ref=sib_rdr_toc/.
3. *Berners-Lee T.* Business Model for the Semantic Web, 2001. — http://www.w3.org/DesignIssues/Business.
4. *OWL*. Web Ontology Language. W3C. — http://www.w3c.org/TR/owl-features/.
5. *Knowledge* level engineering: ontological analysis /Alexander J.H., Freilin M.J., Shulman S.J. et al. // Proc. of the 5h National Conf. on Artificial Intelligence, AAAI, Philadelphia. — 1986. — P. 963–968.
6. *Musen M.* Domain Ontologies in Software Engineering: Use of Protégé with the EON Architecture // Methods of Information in Medicine. — Stutgart. — 1998. — **37**, № 4, 5. — P. 540–550.
7. *Рогушина Ю.В.* Применение методов индуктивного вывода для создания прикладных экспертных систем // Разработка и использование информационных технологий в системах управления: Сб. науч. тр. — Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1993. — С. 122–128.

*Надійшла 18.07.2005*