Список літератури

1. Структурированные данные и семантическая паутина: технологи Wiki / А. Н. Глибовец, Н. Н. Глибовец, Д. Е. Покопцев, М. О. Сидоренко // Проблемы программирования. – 2013. – № 1. – С. 45–67.
2. Cannataro M. Semantics and knowledge grids: Building the next-generation grid / M. Cannataro, D. Talia // IEEE Intelligent Systems. – 2004. – Vol. 19 (1). – P. 56–63.
3. Cannataro M. The knowledge grid / M. Cannataro, D. Talia // Comm. ACM. – 2003. – Vol. 46 (1). – P. 89–93.
4. Mode of access [Електронний ресурс]. – Режим доступу: www.ontogrid.eu – Назва з екрана.
5. Talia D. How Distributed Data Mining Tasks can Thrive as Knowledge Services / D. Talia, P. Trunfio // Comm. ACM. – 2010. – Vol. 53 (7). – P. 132–137.
6. TextGrid Virtual esearch environment humanities [Електронний ресурс]. – Режим доступу: www.textgrid.de – Назва з екрана.
7. Weka4WS – Data Mining Software for Web Services [Електронний ресурс]. – Режим доступу: http://gridlab.dimes.unical.it/weka4ws/about/ – Назва з екрана.
8. Zhuge H. The Web Resourse Spase Model / H. Zhuge – Springer, 2007. – 281 p.
9. Zhuge H. China's E-Science Knowledge Grid Environement [Електронний ресурс]. – Режим доступу: www.computer.org/inelligent (назва з екрана).
10. Zhuge H. Resourse Spase Model, its Design Vtthod and Applications / H. Zhuge // Journal of System and Software. – 2004. – Vol. 72 (1). – P. 71–81.
11. Zhuge H. Semantic component networking: Toward the synergy of static reuse and dynamic clustering of resources in the knowledge grid / H. Zhuge // Journal of Systems and Software. – 2006. – Vol. 79. – P. 1469–1482.
12. Zhuge H. The Knowledge Grid [Електронний ресурс] / H. Zhuge. – Word Scientific, 2004. – 264 p. – Режим доступу: www.books.google.com.ua/books? isbn=9814291773 – Назва з екрана.

N. Glybovets

KNOWLEDGE GRID

In the article are analyzed and summarized current trends Knowledge Grid.

Keywords: knowledge grid, single semantic image, semantic link network.

Матеріал надійшов 01.09.2013

УДК 681.3: 519.68

Жежерун О. П., Мельничук В. М.

ВИКОРИСТАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ РОЗВ'ЯЗКУ ЗАДАЧ УНІФІКАЦІЇ

Задача уніфікації є однією з класичних задач штучного інтелекту, використовується в системах логічного виводу. Традиційний шлях її вирішення потребує великих обсягів обчислень, у загальному випадку – експоненційний. Тому задача пошуку уніфікатора нетрадиційним шляхом, який, можливо, скорочує цей шлях, для наведених класів задач виглядає актуальним.

Використання генетичних алгоритмів дозволяє отримати додатковий спосіб у вирішенні проблеми уніфікації.

Ключові слова: логічне програмування, теорія першого порядку, алгоритм уніфікації, генетичний алгоритм.

1. Основні поняття з уніфікації

Наведемо основні означення з теорії уніфікації [2].

Означення 1.

Термом – це будь-який символ із змінних та константних або ж функціональних символів, які йдуть за послідовністю термів, відокремлених комами, дужками.

Означення 2.

Підстановкою називається відображення σ з множини змінних у множину термів:

$$\sigma : X \to T.$$

Підстановку σ, для якої σ (x) – це g (h (a), a) та σ (y) – це h (a), можна записати як множину: {x/g (h (a), a), y/h (a)}. Застосування підстановки до терму записується як σ (t) та означає терм, в якому всі змінні x_i t замінені на σ (x_i).

Означення 3.

Підстановка *σ* називається *уніфікатором* для множини термів $\{t_1, t_2, ..., t_n\}$ тоді і тільки тоді, коли $\sigma(t_1) = \sigma(t_2) = ... = \sigma(t_n)$. Кажуть, що множина термів $\{t_1, t_2, ..., t_n\}$ *уніфікується*, якщо для неї існує уніфікатор.

Для знаходження уніфікатора двох термів використовують алгоритм уніфікації Робінсона [5]. На вхід функція *unify* приймає два терми *E1*, *E2*. Спочатку терми перевіряються, чи є вони константами або пустими, для цього використовуються допоміжні функції *constants* та *empty*. У випадку рівності термів відбувається вихід з функції *unify* та, як значення, повертається порожня підстановка. Інакше отримуємо значення *FAIL*, що свідчить про неможливість уніфікувати два дані терми. Оскільки алгоритм Робінсона працює за допомогою рекурсії, то перша умова використовується для виходу з рекурсії. Далі проходить перевірка, чи є один з термів змінною, якщо ця умова є істиною, то відбувається перевірка, чи входить терм-змінна до іншого терму, це робиться за допомогою функції *occurs*, якщо умова виявилась вірною, то відбувається вихід з результатом *FAIL*, ця умова дає можливість уникнути безкінечної уніфікації. Якщо ж терм-змінна не трапляється в іншому термі, то відбувається підстановка терма замість терму-змінної. Якщо ж виявилось, що тільки один з термів порожній, то відбувається вихід з результатом *FAIL*. Потім відбувається виділення перших елементів обох термів за допомогою функції *head* та рекурсивного застосування до них функції *unify*. Результат функції записується у змінну SUB1, вона і є першою підстановкою. Якщо результат роботи функції *unify* задовільний, то підстановка застосовується до першого і другого терму за допомогою функції *apply* та виконується рекурсивний запуск функції уніфікації від неуніфікованих частин термів. Результуюча підстановка формується композицією (добутком) усіх попередніх підстановок за допомогою функції *composition*.

За [4] можна виділити клас термів, для яких алгоритм уніфікації Робінсона працює неефективно, а саме: він є експоненційним як за часом, так і за витратами пам'яті. У загальному випадку пара термів записується у такій формі:

$$h(x_1, x_2, ..., x_n, f(y_0, y_0), ..., f(y_{n-1}, y_{n-1}), y_n);$$
$$h(f(x_0, x_0), f(x_1, x_1), ..., f(x_{n-1}, x_{n-1}), y_1, ..., y_n, x_n).$$

Позначимо ці терми як *h-терми*.

Уніфікація цих двох термів породжує уніфікатор з такими властивостями: кожна змінна x_i та y_i пов'язана з термом, у якого $2^{i+1}-1$ символів. Проблема у тому, що підстановки містять багато копій однакових підтермів.

2. Розробка генетичного алгоритму для уніфікації термів 1-го порядку

Класичний генетичний алгоритм складається з таких кроків [3]:

1. формування початкової популяції,
2. оцінка особин популяції,
3. відбір особин для схрещування,
4. схрещування,
5. мутація,
6. формування нової популяції,
7. якщо популяція не є кінцевою, то повертаємось до кроку 2. В іншому випадку – зупинка алгоритму.

Кросовер (кросинговер) – операція, в результаті якої дві хромосоми обмінюються своїми частинами.

Мутація – випадкова зміна одного чи кількох генів у хромосомі.

Інверсія – зміна порядку слідування генів у хромосомі або в її частині.

Функція пристосованості – функція оцінки, вона показує міру пристосованості даного індивіда в популяції. Відіграє важливу роль, оскільки дозволяє оцінити рівень пристосованості конкретних індивідів у популяції та вибрати найбільш пристосованих.

Для формування початкової популяції потрібно визначити, як будуть представлятися особини. Особиною популяції буде одна підстановка, яка є потенційним уніфікатором.

Особина складається з однієї хромосоми, а хромосома – з генів. Геном є відображення однієї змінної у терм. Довжина хромосоми дорівнює кількості змінних у початкових термах, тобто потужність множини змінних термів.

Як приклад наведемо два терми:

$$t_1 = f(x, a), \; t_2 = f(y, z).$$

Множина змінних, які зустрічаються у t_1 та t_2:

$$\{x, y, z\}.$$

Отже, хромосома буде складатися з трьох генів.

Лема 2.1. Час роботи ГА залежить від кількості змінних, які входять до початкових термів.

Доведення:

При збільшенні кількості змінних, які входять у терми, зростає довжина хромосоми. Це призводить до того, що зростає час обрахунку функції пристосованості для одного індивіда. Тому час роботи алгоритму зростає.

Як бачимо на прикладі, порядок генів має значення і залежить від змінних, це визначає *локус* гена. Тепер потрібно визначити алель.

Алель може приймати значення констант, змінних та інших підтермів з термів у t_1 та t_2. Наприклад: *{a, x, y, z, f (x, a), f (y, z), f (a, a), f (f (x, a), x), z), ...}* .

Бачимо, що кількість потенційних розв'язків нескінченна навіть для досить простих термів. Тому потрібно побудувати певний механізм породження підстановок для ГА.

Лема 2.2. Кількість поколінь прямо пропорційна з кількістю змінних t_1 і t_2 з певним коефіцієнтом *l*, який більше одиниці.

Доведення:

Нижньою межею кількості поколінь є кількість змінних у t_1 і t_2. Тому кількість поколінь не менша за *n*. При використанні механізму мутації можливе виродження популяції, тому для усунення виродження збільшують кількість поколінь для знаходження уніфікатора.

3. Механізми породження підстановки

Для роботи генетичного алгоритму потрібно визначити множину, з якої будуть генеруватися особини. Це дуже важливий етап як для початку роботи генетичного алгоритму (під час створення початкової популяції), так і у процесі роботи алгоритму. Пошук розв'язку повинен проходити у множині, де розв'язок може бути знайдено, та виключати підмножини, які не містять потенційних розв'язків.

Використовуючи вищезгадані терми t_1 і t_2, зробимо підстановку всіх змінних на константу *a*:

$$\{x/a, y/a, z/a\} .$$

Тоді терми набувають такої форми:

$$t_1 = f(a, a),\ t_2 = f(a, a).$$

Отже, підстановка *{x/a, y/a, z/a}* буде уніфікатором термів t_1 і t_2.

Проаналізувавши даний приклад і подібні, робимо висновок, що для уніфікації термів першого порядку підстановки можна вибрати з множини універсуму Ербрана.

Переваги використання універсуму Ербрана:

- генерація елементів універсуму Ербрана – це нескладна операція, вона подібна до генерації комбінаторних об'єктів (перестановок);
- покрокова побудова універсуму добре підходить до моделі роботи генетичного алгоритму;
- універсум Ербрана – це зліченна множина, тому можна побудувати однозначне кодування з елементів універсуму у множину натуральних чисел.

За [1] означення ербранівського універсуму має такий вигляд:

Нехай *S* – множина диз'юнктів і H_0 – множина констант, які зустрічаються в *S*. Якщо жодна константа не зустрічається в *S*, то H_0 складається з однієї константи, $H_0 = \{a\}$. Для $i = 1, 2, \ldots$ множина H_{i+1} являє собою об'єднання H_i і множини всіх термів, що мають вигляд $f(t_1, t_2, \ldots, t_n)$ (при всіх *n*) для всіх функцій *f* арності *n*, що зустрічаються в S, де $t_j \in H_i$ ($j = 1, 2, \ldots, n$). Тоді кожна множина H_i називається множиною констант *i*-го рівня для *S*, і H_∞ називається ербранівським універсумом *S*.

У загальному випадку універсум Ербрана (УЕ) можна будувати нескінченно, тому для практичних задач використаємо тільки декілька рівнів. Одним з критеріїв вибору підстановки для схрещування буде довжина підстановки, яка у свою чергу залежить від рівня H_i.

4. Вибір та визначення функції пристосованості

Однією з основних частин генетичного алгоритму є функція пристосованості. Вибір функції пристосованості залежить від швидкості обрахунку функції пристосованості, вона повинна бути якомога мінімальною. Для визначення пристосованості підстановок можна підставити їх у кожний з вхідних термів і визначити ступінь подібності змінених термів.

Означення 4.1. Відстань Хеммінга.

Функція пристосованості вираховується як різниця між двома термами, для яких проведено операцію підстановки з можливого уніфікатора:

$$d(\sigma(t_1), \sigma(t_2)),$$

де d – це відстань Хеммінга, σ – підстановка, t_1, t_2 – терми.

Оскільки під час роботи алгоритму (особливо операції мутації та схрещування) довжини особин змінюються, то відстань Хеммінга не відповідає загальному випадку генетичного алгоритму. Проте вона може бути застосованою у специфічних задачах, де довжини особин однакові під час усієї роботи алгоритму.

Означення 4.2. Відстань Левенштейна.

Нехай *u* та *v* – два рядки з довжинами M та N відповідно, тоді відстань Левенштейна d (S_1, S_2) можна підрахувати за допомогою рекурентної формули:

$$d(u[1..i], v[1..j]) = \min \begin{cases} \mathrm{d}(u[1..i], v[1..j-1])+1 \\ \mathrm{d}(u[1..i-1], v[1..j])+1 \\ \mathrm{d}(u[1..i-1], v[1..j-1])+\delta_{u[i],v[j]} \end{cases},$$

де $\delta_{u[i], v[j]}$ дорівнює нулю при $u[i] = v[j]$ і одиниці в іншому випадку.

Відстань Левенштейна підходить для порівняння рядків різної довжини. Це відповідає генетичному алгоритму в загальному вигляді.

Метою роботи алгоритму є пошук такої підстановки, для якої відстань Левенштейна буде мінімальною, тобто розв'язується задача мінімізації.

Які можливі додаткові критерії для визначення пристосованості індивідів?

Як приклад, візьмемо два терми:

$$t_1 = f(x),\ t_2 = f(y).$$

З УЕ для уніфікації термів t_1, t_2 можна вибрати такі підстановки:

$$\sigma = \{x / a, y / a\},\ \delta = \{x / f(a), y / f(a)\}.$$

Довжини уніфікаторів будуть різними, тому ще одним критерієм, який характеризуватиме пристосованість індивіда, буде кількість символів, яка входить у підстановку, тобто довжина підстановки.

Отже, знаходження індивідів, для яких відстань Левенштейна буде мінімальною, та вибір серед них індивіда з найменшою довжиною підстановок – буде основною задачею роботи алгоритму.

5. Опис роботи алгоритму

Вхід: Два терми t_1, t_2.

1. Аналіз термів: виділення списків констант, змінних, функціональних символів.
2. Генерування УЕ за константами, виділеними на кроці 1, з глибиною, яка забезпечить унікальну підстановку для кожної змінної.
3. Створення початкової популяції підстановок: з УЕ випадковим чином вибираємо значення генів для індивідів. Розмір популяції залежить від кількості змінних, які було виділено на кроці 1.
4. Визначення ступеня пристосованості: підраховуємо відстань Левенштейна та довжини отриманих підстановок.
5. Сортуємо популяцію у порядку зростання відстані Левенштейна та довжини підстановок.
6. Виконуємо відбір турнірним методом індивідів для схрещування.
7. Виконуємо схрещування, для чого випадковим чином знаходимо точку розриву двох хромосом і здійснюємо обмін генами, утворених індивідів додаємо до популяції.
8. Виконуємо мутацію для випадковим чином вибраних особин. Рівень мутації задається у відсотках, наприклад 20 %.
9. Підраховуємо пристосованість індивідів.
10. Вибираємо індивідів за рівнем пристосованості; береться така сама кількість, як і у початковій популяції.
11. Перевіряємо чи не дорівнює відстань Левенштейна нулю, якщо так, то вибираємо підстановки з найменшою довжиною, якщо ні – повторюємо з кроку 6. Зупинка відбувається також у випадку досягнення максимального числа генерацій.

6. Теоретичні характеристики швидкодії ГА

При роботі алгоритму Робінсона для уніфікації h-термів основною операцією, яка постійно виконується, є суперпозиція підстановок. Кількість суперпозицій залежить від кількості змінних, які входять в h-терм, та зростає експоненційно. Притому довжина уніфікатора також зростає експоненційно.

Натомість, під час роботи ГА підстановки отримуються не з суперпозиції вже отриманих підстановок, а беруться з УЕ, який поступово генерується залежно від кількості змінних, що входять у початкові терми.

Тому основний виграш у швидкості роботи генетичного алгоритму отримаємо за рахунок використання елементів УЕ, а не використання операції суперпозиції підстановок.

Лема 6.1. Час роботи генетичного алгоритму визначається за формулою:

$$t(n) = kT(n),$$

де T (n) – час роботи алгоритму Робінсона, t (n) – час роботи генетичного алгоритму, n – кількість змінних, що входять у терми, k – коефіцієнт, який вказує на пришвидшення роботи ГА:

$$0 < k \le 1.$$

Доведення:

Під час роботи алгоритму уніфікації Робінсона, формується уніфікатор для h-термів. Однією з особливостей уніфікатора h-термів є те, що довжина уніфікатора збільшується експоненційно. Оскільки метою роботи алгоритмів уніфікації є знаходження уніфікатора, то час їх роботи буде пропорційним. Коефіцієнт пропорційності залежить від порівняння швидкодії операції суперпозиція для алгоритму Робінсона та операції генерації елементів УЕ для генетичного алгоритму.

Враховуючи вищезгадані леми, можна сформулювати теорему, яка буде характеризувати генетичний алгоритм уніфікації.

Теорема 6.1.

Якщо терми t_1 і t_2 мають уніфікатор, то:

1) його точно можна знайти;
2) для знаходження уніфікатора потрібно не більше m кроків генетичного алгоритму;

3) швидкість роботи не гірше ніж у звичайного алгоритму (пришвидшення визначається константою k з леми 6.1).

Інакше алгоритм доходить до максимальної кількості поколінь та завершує свою роботу.

Доведення:

Пункт 1, уніфікатор знаходиться у підмножині УЕ, тому ГА, який працює з елементами УЕ, обов'язково знайде уніфікатор.

Пункт 2 можна отримати з леми 2.2, а число *m* залежить від кількості змінних у t_1 і t_2.

Пункт 3 отримуємо з леми 6.1.

7. Експериментальна робота та оцінки роботи ГА уніфікації

Для визначення ефективності роботи ГА уніфікації проведено ряд експериментів, під час яких було визначено швидкодію уніфікації. Класом термів для уніфікації були h-терми з різною кількістю змінних.

Для порівняння ефективності роботи було реалізовано алгоритм уніфікації Робінсона та визначено його швидкодію для уніфікації h-термів.

Тестування проводилося за допомогою таких апаратних та програмних засобів:

Апаратне забезпечення:

- процесор: Intel Atom N2600 Dual Core 1.6GHz;
- пам'ять: DDR3 2GB.

Програмне забезпечення:

- операційна система: Windows 7 Professional 32-bit;
- версія інтерпретатора Python: Python 2.7.3.

Час роботи алгоритмів вимірюється за допомогою вбудованої функції time. Також була спроба використати модуль Python cProfile, але вивід цього модуля є дуже детальним, що ускладнює побудову графіків залежності часу роботи від кількості змінних у h-термі.

8. Визначення швидкості роботи алгоритму уніфікації Робінсона для h-термів

Спеціально побудовані h-терми з великою кількістю змінних мають просту, і водночас громіздку будову, тому для усунення можливості помилки реалізовано процедуру автоматичної побудови h-термів заданої розмірності.

Для порівняння роботи ГА з роботою алгоритму Робінсона реалізовано алгоритм мовою Python. Він представляє класичний алгоритм уніфікації, де можна виділити такі особливості реалізації:

- терми записуються за допомогою списків;
- підстановки записуються за допомогою словника;
- композиція підстановок реалізована за допомогою вбудованого у тип словник методу update;
- реалізації функції підстановки (*substitute*) та функції входження змінної у терм (occurs_check) є рекурсивними;
- результатом роботи є словник-уніфікатор або рядок "Fail".

Було побудовано 20 h-термів, довжина від 1 до 20. Також проведено 20 тестів для кожної з довжин h-термів та визначено середній час роботи алгоритму.

Графік залежності середнього часу роботи уніфікації Робінсона від кількості змінних, які входять в h-терм, має експоненційний характер (рис. 1).

9. Визначення швидкості роботи генетичного алгоритму уніфікації для h-термів

Параметри генетичного алгоритму:

1. розмір початкової популяції: 20 особин;
2. турнірний відбір для схрещування;
3. рівень мутації: 10 %;
4. функції пристосованості: відстань Левенштейна, довжина підстановки;
5. кількість тестів для одного h-терму: 20.

Параметри підібрано експериментальним шляхом, вони відображають час роботи, при якому швидкодія алгоритму максимальна (рис. 1).

Графік виявляється схожим на графік для алгоритму Робінсона, але при безпосередньому співставленні графіків можна побачити відмінність (рис. 1).

У порівнянні роботи двох алгоритмів використовуються такі дані:

- кількість змінних у термах: від 1 до 20;
- кількість тестів для кожного терму: 20.

Графік роботи алгоритму Робінсона піднімається круто вгору, а графік робота генетичного алгоритму піднімається полого.

Характер графіків зумовлено тим, що результативний уніфікатор зростає експоненційно, тому генетичний алгоритм уніфікації не змінює характер складності, але робить криву зростання часу доволі пологою.

Для точніших аналітичних висновків складено порівняльну таблицю залежності часу роботи алгоритмів (таблиця).

Проаналізувавши таблицю, можна зробити висновок, що різниця у часі роботи алгоритмів відрізняється майже у 3 рази. Тому експеримен-

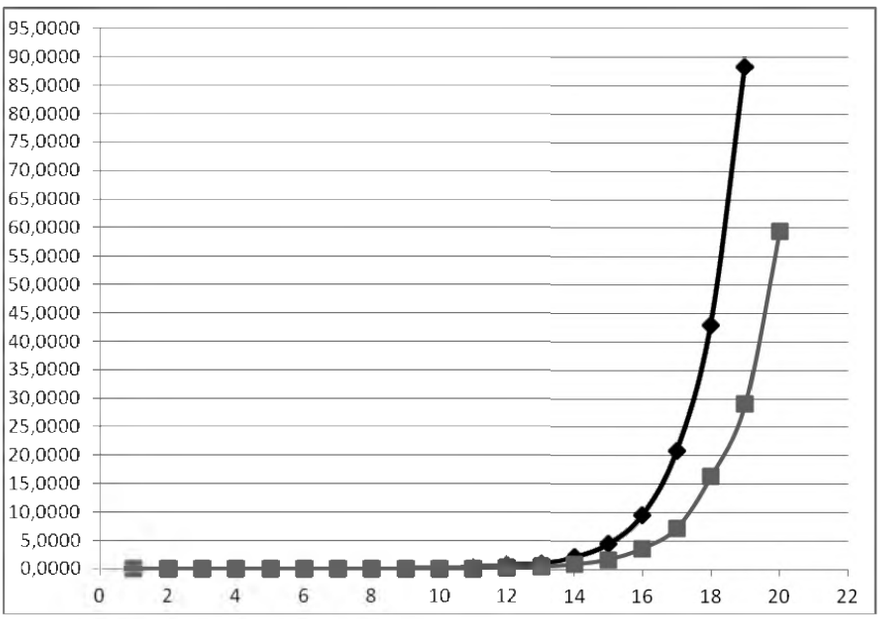

Рис. 1. Графіки швидкодії роботи алгоритмів уніфікації

тальним шляхом доведено, що ми отримали приріст у продуктивності, який зменшив час виконання алгоритму на дві треті.

Якщо позначити за T (n) час виконання алгоритму уніфікації Робінсона, а за t (n) час виконання ГА уніфікації, то можна записати таку рівність:

$$t(n) = \frac{1}{3}T(n).$$

Наступна формула співпадає з формулою, що наведена в пункті 2.6. Експериментально виявлено, що коефіцієнт пришвидшення k=0,3. Це співпадає з *Лемою 6.1*.

Зважаючи на час роботи алгоритму, можна зробити висновок, що *Лема 2.1* підтверджується експериментально.

Вищезгадані дані з Таблиці ілюструють *Теорему 6.1*.

Висновки

У цій статті розглянуто спосіб вирішення задачі уніфікації з використанням генетичного алгоритму. Крім того, розглянуто існуючі методи уніфікації термів 1-го порядку.

Проведено порівняння роботи уніфікації для термів з різною кількістю змінних і визначено можливість застосування генетичного алгоритму уніфікації.

Як приклад ефективності роботи генетичного алгоритму наведено клас задач, для яких

Таблиця. **Порівняння часу роботи алгоритмів уніфікації**

К-ть змінних	Час роботи алгоритму Робінсона	Час роботи генетичного алгоритму	Довжина уніфікатора
1	0,00099992752	0,00100016594	6
2	0,00099992752	0,00100026594	44
3	0,00200009346	0,00100036594	134
4	0,00399994850	0,00199985504	328
5	0,00999999046	0,00300002098	730
6	0,01399993896	0,00400018692	1548
…	…	…	…
14	2,13200020790	0,88299989700	425756
15	4,44400000572	1,74699997902	851726
16	9,57800006866	3,70700001717	1703680
17	20,79600000380	7,28999996185	3407602
18	43,00099992750	16,31399989130	6815460
19	88,41799998280	29,04699993130	13631190
20	185,67300009700	59,41899991040	27262664

звичайний алгоритм уніфікації (алгоритм Робінсона) є неефективним.

Результатом статті є алгоритм уніфікації, який працює на базі генетичних алгоритмів, та сформульовано теорему, яка описує застосовність ГА.

За результатами тестування генетичного алгоритму уніфікації можна зробити висновок, що він працює у 3 рази швидше, ніж звичайний алгоритм уніфікації для класу задач h-термів.

Список літератури

1. Капітонова Ю. В. Основи дискретної математики / Ю. В. Капітонова. – К. : Наук. думка, 2002. – 579 с.
2. Кривий С. Л. Дискретна математика: Вибрані питання / С. Л. Кривий. – К. : Видавничий дім «Києво-Могилянська академія», 2007. – 572 с.
3. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский // Горячая линия. – Телеком, 2006.
4. Baader F. Unification Theory / Franz Baader, Wayne Snyder // Handbook of Automated Reasoning. – Elsevier Science Publishers, 2001.
5. Luger G. Artificial Intelligence: Structures and Strategies for Complex Problem Solving / George F. Luger. – Addison-Wesley, 2008.

O. Zhezherun, V. Melnichuk

THE USE OF GENETIC ALGORITHMS FOR DECISION OF UNIFICATION TASKS

A task of unification is one of classic task of artificial intelligence, used in the inference systems. The traditional way needs the large volumes of calculations, in general case of exponent scale. Therefore task of searching of unificator by an unconventional way looks actual.

The use of genetic algorithms allows to get an additional method in the decision of problem of unification.

Keywords: logical programming, first-order theory, algorithm of unification, genetic algorithm.

Матеріал надійшов 14.06.2013

УДК 004.89

Афонін А. О., Лялецький О. В.

ОСОБЛИВОСТІ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ІНФОРМАЦІЇ У СУЧАСНИХ СИСТЕМАХ АВТОМАТИЗАЦІЇ МІРКУВАНЬ

У статті аналізуються сучасні парадигми та стилі інтелектуальної обробки інформації в системах автоматизації міркувань.

Ключові слова: автоматизація міркувань, обробка інформації, машинне доведення, Theorema, Isabelle, Coq, Automath, Lambda-Clam, Mizar, SAD.

Вступ

У наш час посилилась увага до створення нових систем автоматизації міркувань як на базі інтеграції вже наявних інтелектуальних сервісів, так і за рахунок розвитку підходів до їх створення з урахуванням сучасних досягнень в області інформатики та інформаційних технологій.