ПРО ОДИН ПІДХІД ПОБУДОВИ СИСТЕМ ЦЕНТРАЛІЗОВАНИХ БАЗ ЗНАНЬ, ЯКІ ФОРМУЮТЬСЯ МЕТОДАМИ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

М.М. Глибовець, кандидат фізико-математичних наук, доцент

Національний університет "Києво-Могилянська Академія",
м. Київ, вул. Сковороди, 2, корпус 1,
т. 463-69-85, glib@ukma.kiev.ua

С.В. Адамчук, аспірант

Київський національний університет ім. Тараса Шевченка,
03680, м. Київ-680, проспект академіка Глушкова 2, корпус 6,
т. 259-04-27, tiex@ukr.net

У даній публікації ми розглядаємо можливості значного прискорення формування бази знань методом навчання з підкріпленням за рахунок використання однієї й тієї ж бази знань багатьма клієнтами (агентами) одночасно. У роботі описано власну розробку: сервер баз знань Adapton Server.

In this paper we explore the way of increasing of Reinforcement Learning process when the same knowledgebase is used by many agents simultaneously. Our application: Adapton Server – the server of knowledgebases (with the network access) is described here.

Вступ

Базу знань, сформовану за допомогою методів навчання з підкріпленням (Reinforcement Learning) функціонально можна представити як множину можливих станів середовища та оцінки різних можливих дій агента у цих станах. На основі таких оцінок агент вибирає певну дію для прийняття рішення, самі ж оцінки формуються на основі попереднього досвіду агента. Таким чином у кожен момент часу активно використовується агентом певний невеликий фрагмент бази знань (поточний стан середовища). В той же час інші, аналогічні агенти, могли б використовувати (і навіть модифікувати) інші фрагменти цієї ж бази знань. У багатьох випадках такі централізовані бази знань можуть бути досить корисними, зокрема це може значно прискорити процес навчання.

Терміни "агент" та "середовище" розуміються тут у тому значенні, яке в них вкладається у класичних працях про навчання з підкріпленням, наприклад [1]. Зокрема, під агентом розумітимемо елемент, який безпосередньо взаємодіє з середовищем: одержує інформацію із зовнішнього середовища, впливає на це середовище та одержує числові оцінки своїх дій. При такому виборі дій агент як керується інформацією з бази знань, так і модифікує базу знань на основі своїх оцінок.

Нашим завданням буде розробка програмного забезпечення для централізованого зберігання бази знань, та забезпечення доступу до бази знань з віддалених комп'ютерів. При цьому ми повинні забезпечити безконфліктний одночасний доступ багатьом клієнтам до однієї й тієї ж бази знань. У цій публікації ми опишемо реалізацію такого типу сервера баз знань та продемонструємо приклад створення клієнтського програмного забезпечення.

Існує багато прикладів реалізації централізованих баз знань, але вони мають ті чи інші вади. Тому для ефективної реалізації таких баз знань ми спробували розробити сервер баз знань Adapton Server, який має клієнт-серверну архітектуру. У цій програмі для використання та накопичення досвіду бази знань використовують теорію навчання агента із підкріпленням [1] і алгоритми формування та оптимізації бази знань описані в [2].

Методи навчання з підкріпленням (Reinforcement Learning) [1] дозволяють формувати базу знань при одночасному розв'язуванні агентами задач (на основі числових оцінок), а також дають можливість безконфліктного паралельного використання єдиної бази знань багатьма агентами, а основне – значно прискорюють навчання, завдяки збільшенню інтенсивності потоку досвіду від багатьох клієнтів.

Сам сервер баз знань та API для написання клієнтського програмного забезпечення розроблені на базі технології MS Framework .Net [6].

Одночасний доступ багатьох клієнтів до єдиної бази знань

Нагадаємо, що загальна схема клієнт-серверної архітектури інтелектуальних баз знань має вигляд представлений на рис. 1 [3, 4].

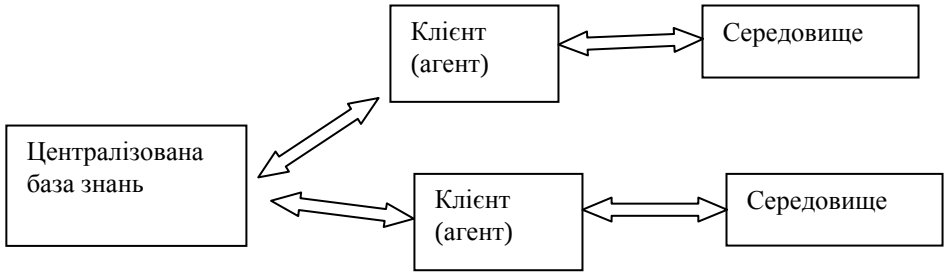

Рис. 1

Розглянемо детальніше засоби розпаралелювання доступу клієнтів до сервер баз знань AdaptonServer.

Тут централізована база знань „консультує" клієнтів щодо поведінки у середовищі. Хоча кожен клієнт взаємодіє із своїм середовищем і нічого „не знає" про інших клієнтів, проте для всіх клієнтів, що під'єдналися до однієї і тієї ж бази знань, середовища повинні бути еквівалентними (детальніше про середовища див. [1, 2]).

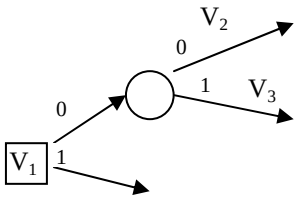

Рис. 2

Взаємодія між агентом та середовищем відбувається на основі дискретного перетворювача. Агент взаємодіє із середовищем послідовно надсилаючи та одержуючи фіксовані сигнали (символи із деякого фіксованого алфавіту A). Таким чином для кожного клієнтського під'єднання у виділений момент часу сервер бази знань зберігає деяку маску, що ідентифікує певний стан (ідентифікатор). Одержуючи символ із середовища агент передає його серверу, відповідно до цього змінюється поточний стан даного клієнта. Аналогічно, коли сервер очікує символ від агента (прийняття рішення), агент звертається за рішенням до централізованої бази знань, яка формує рішення для даної ситуації – випробуване на основі досвіду багатьох клієнтів.

Загалом, як показувалось вище, паралельне використання єдиної бази знань багатьма клієнтами не представляє ніяких проблем. Структура переходів при цьому не змінюється, а змінюються лише ціни станів (чи переходів) – які є основними факторами для вибору рішення, проте у межах незначного проміжку часу вони змінюються не дуже сильно.

Більш складна ситуація виникає тоді, коли робиться оптимізація бази знань: злиття еквівалентних пар станів [2]. Це призводить до зміни структури бази знань і зникнення деяких станів, що могло б призвести до ситуації, коли ідентифікатор стану, асоціативний із деяким клієнтом може стати неактуальним. Щоб подолати цю проблему, необхідно при кожному злитті станів з ідентифікаторми S_1 та S_2 (S_2 переіменовується в S_1) переглянути також і список ідентифікаторів станів під'єднань клієнтів і замінити всі ідентифікатори S_2 на S_1. Оскільки процес оптимізації запускається адміністратором окремим потоком, може бути так, що в цей момент клієнти звертаються до методів `SendSymbol`, `ReceiveSymbol` чи `SendRate`, яким повинно гарантуватись незмінність поточного стану у базі знань, для цього у цих методах використовуються засоби C# для тимчасового блокування доступу до об'єкту (бази знань) іншим потокам – зарезервована конструкція `lock`, продемонструємо її використання наступними прикладами.

```
/// Надійшов новий символ із середовища
public void SendSymbol(byte symbol)
{
...
        // блокуємо базу знань, щоб не змінився іншим потоком ідентифікатор стану
        lock(_knowledgebase)
        {
                // сервер аналізує символ, одержаний від клієнта (середовища)
                // та переходить у наступний стан
```

```
        }
...
}
/// Одержуємо символ з бази знань
public byte ReceiveSymbol()
{
...
        // блокуємо базу знань, щоб не змінився іншим потоком ідентифікатор стану
        lock(_knowledgebase)
        {
                // сервер формує символ для передачі клієнту
                // та переходить у наступний стан
        }
...
}
/// Модифікується база знань згідно з оцінкою середовищем
public void SendRate(double rate)
{
...
        // блокуємо базу знань, щоб не змінився іншим потоком ідентифікатор стану
        lock (_knowledgebase)
        {
                // сервер аналізує оцінку середовищем дій агента і модифікує базу знань
                // та переходить у наступний стан
        }
...
}
```

Розробка клієнтського програмного забезпечення

Запропонована фірмою Microsoft технологія розробки ПЗ на платформі .Net поєднує в собі максимальну простоту при розробці та максимальну ефективність програмного коду. Технологія Remoting.Net забезпечує зручну і в той же час безпечну взаємодію між різними програмами (в тому числі і на різних комп'ютерах) [5]. Програмний комплекс Adapton Server також включає API для написання клієнтського програмного забезпечення з використанням механізму .Net Remoting.

Для написання клієнтів до сервера баз знань використовується бібліотека з проксі-класом, яка відкриває з'єднання до бази знань та працює із цим з'єднанням. Перед початком роботи із з'єднанням клієнтська програма створює екземпляр класу з'єднання AdaptonConnection (при цьому на клієнті створюється екземпляр проксі-об'єкта, методи якого фактично виконуються на сервері).

Специфікація класу для написання програмного ПЗ для доступу до сервера баз знань AdaptonConnection:

```
Відкриваємо базу знань за її іменем
public void Open(string KnowledgeBaseName, string UserName, string Password)
Відкриваємо базу знань за індексом
public void Open(int idx, string UserName, string Password)
Одержуємо новий символ із середовища
public void SendSymbol(byte symbol)
Одержуємо символ з бази знань
public byte ReceiveSymbol()
Надсилається оцінка середовищем агенту
public void SendRate(double rate)
Зкидування зв'язку. Перехід у початковий стан.
public void Reset()
Закривається під'єднання до бази знань
public void Close()
```

Приклад використання AdaptonConnection у клієнтській програми:

```
...
connection = new CommonKnowledge.Adapton.Connection.AdaptonConnection();
connection.Open(System.Configuration.ConfigurationSettings.AppSettings["KnowledgeBaseName"],
"UserName", "Password");
...
connection.SendSymbol(0);
byte symb = connection.ReceiveSymbol();
connection.Close();
```

Висновки

У цій роботі продемонстровано переваги реалізації інтелектуальних баз знань за рахунок використання сучасного інструментарію розробки – Microsoft .Net, та використання спеціалізованих методів формування баз знань шляхом навчання з підкріпленням при доступі до єдиної бази знань багатьма клієнтами одночасно. Зазначимо, що запропонована методологія створення баз знань централізованого типу значно впливає на підвищення швидкості формування бази знань (навчання) за рахунок можливості паралельної корекції єдиної

бази знань, а швидкість навчання –на сьогодні є однією із найактуальніших проблем при розробці інтелектуальних систем будь-якого типу.

Література

1. Sutton R., Barto A. Reinforrcement Learning // MIT Press, Cambridge, MA, 1998 A Bradford Book
2. Адамчук С.В., Адаптивне виявлення особливостей середовища та знаходження оптимальної поведінки інтелектуального агента в цьому середовищі // Вісник Київського університету. №1, 2004.
3. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. - СПб.: Питер, 2001. – 384 с.: ил.
4. Гасанов Э.Э., Кудрявцев В.Б. Теория хранения и поиска информации. - М.: Физматлит, - 2002. - 228 с.
5. Майо, Дж., C#: Искусство программирования. Энциклопедия программиста: Пер. с англ. – СПб.: ООО “ДиаСофтЮП”, 2002. – 656 с.
6. Платт, Д., Знакомство с Microsoft.Net/Пер. с англ. – М.: Издательско-тороговый дом “Русская Редакция”, 2001. – 240 с.: ил.