Проаналізовані сучасні підходи до визначення концепції агента, обґрунтовані вибір та модифікація алгоритмів оптимізації прийняття рішення інтелектуальним агентом, наведені структура програмного комплексу моделювання та результати експериментів

Ключові слова: інтелектуальний агент, раціоналізація, самонавчання, логічне виведення, процес прийняття рішення

Проанализированы современные подходы к определению концепции агента, обоснованы выбор и модификация алгоритмов оптимизации принятия решения интеллектуальным агентом, приведены структура программного комплекса моделирования и результаты экспериментов

Ключевые слова: интеллектуальный агент, рационализация, самообучение, логический вывод, процесс принятие решения

Article there were analyzed current approaches to define the concept of agent, reasoned the choice and modification of the algorithms' decision-making optimization by the intelligent agents, given the structure of simulation software and experiments results

Keywords: intelligent agent, rationalization, self-learning, logical conclusion, decision-making process

УДК 004.832.2

МОДЕЛЮВАННЯ ДІЙ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ В СЕРЕДОВИЩІ ЗІ ЗМІННИМИ ХАРАКТЕРИСТИКАМИ

С.І. Шаповалова
Кандидат технічних наук, доцент
Кафедра автоматизації проектування енергетичних процесів і систем*
Контактний тел: (044) 406-83-72, 067-757-36-92
E-mail: lana@aprodos.ntu-kpi.kiev.ua

Д.І. Третьяков*
*Національний технічний університет України „Київський політехнічний інститут"
пр. Перемоги, 37, корп.5, м. Київ, 03056
Контактний тел: 093-937-76-34
E-mail: dimaty@bigmir.net

Для сучасних агентних систем дедалі частіше постає необхідність оптимізації поведінки згідно зовнішнього середовища, що змінюється, а також здатності до накопичення та аналізу досвіду. Крім того, для подібних систем особливо гостро стоїть питання коректності їх поведінки. Це певною мірою входить у конфлікт з жорсткими обмеженнями на час реакції і обчислювальною потужністю обладнання. Тому проблема оптимізації поведінки автономного агента в середовищі зі змінними характеристиками є актуальною.

Постановка проблеми – створення програмного забезпечення процесу прийняття рішення інтелектуальним агентом в мультиагентній системі для вирішення поставленої цілі.

Метою статті є представлення математичної моделі логічного агента, що має характеристики раціональності, мобільності, мультиагентності та самонавчання для функціонування в стохастичному середовищі з частковим спостереженням.

Аналіз останніх досліджень

Агентно-орієнтований підхід використовується у відкритих, складних та інтерактивних системах [1]. Основними галузями промисловості, в яких розповсюджені мультиагентні системи, являються автоматизація управління (ARCHON, YAMS, OASIS), збір та обробка інформації (мережеві павуки та Web-роботи) та ігри (ігрові боти).

Базові методи проектування та реалізації логічних агентів описано в [2]. Вони вирізняються досить чітким та нескладним в реалізації математичним описанням й чудово підходять для агентів, базу знань яких можна представити у вигляті пропозиційної логіки та логіки першого порядку. Проте вони не враховують стохастичність властивостей середовища.

Дані недоліки в своїх роботах розглядають Вулдрідж зі співавторами [3 – 6]. Проте моделі, що пропонуються, як правило, розглядають окремі властивості самого агента. При поєднанні властивостей агента в одному програмному модулі постають проблеми, пов'язані з обмеженістю комп'ютерних ресурсів та часу для пошуку рішення. На приклад, в реалізації агентів для реальних бойових симуляторів використовуються алгоритми, що враховують тактико-технічні характеристики військової техніки. Навіть на сучасних потужних комп'ютерах потрібне п'ятихвилинне очікування ходу такої програми [7].

1. Визначення агента та середовища

В загальному випадку агентом вважається система, що здатна адекватно реагувати на зміни зовнішнього

середовища, не передбачені явно його поведінковими механізмами. Ми ж будемо розглядати агента як програмну сутність, здатну діяти в інтересах досягнення цілей, поставлених перед ним користувачем. Показниками «якості» агента є: раціональність, мобільність, мультиагентність, здатність до навчання, автономність, активність.

Початковим етапом проектування агента є вивчення середовища, в якому він має функціонувати. Чим більш обмеженим є середовище, тим простіша задача проектування. Різноманітність варіантів проблемного середовища, які можуть виникати в прикладних задачах, досить велика.

Та існує невелика кількість критеріїв, за якими можна класифікувати ці варіанти. Середовище може бути [2, 8]:

- детермінованим або стохастичним;
- епізодичним (історія виконаних дій агентом не впливає на середовище) або послідоним;
- статичним або динамічним;
- дискретним або безперервним;
- одноагентним або мультиагентним;
- спостерігатися повністю або частково.

Будемо розглядати середовище, що має властивості послідовності, динамічності, безперервності, мультиагентності або одноагентності, з частковим спостереженням, та знаходитися в детермінованому або стохастичному стані в різні моменти часу.

Розіб'ємо всю задачу створення інтелектуального агента для визначеного середовища на окремі підзадачі, а саме:

- вибір оптимального рішення задачі в конкретний момент часу;
- підтвердження або спростування гіпотези можливої дії агентом;
- самонавчання агента при зміні вастивостей агента.

2. Багатошаровий агент

Існує багато підходів до реалізації інтелектуального агента, кожний з яких володіє своїми перевагами та недоліками. Основна ідея багатошарової архітектури полягає в тому, що різні підходи можна поєднувати в єдиній системі завдяки введенню *архітектурних шарів (layers)* [9], кожен з яких вирішує свій спектр завдань найбільш ефективним для нього способом.

У цілому багатошарові архітектури можна розділити на два великих класи:

- *Горизонтальні* – коли кожен з шарів безпосередньо підключений до вхідної інформації про сприйняття навколишнього середовища і до виходу для прийнятих рішень;
- *Вертикальні* – коли в напряму і до вхідних, і до вихідних даних безпосередньо підключаються не більше одного архітектурного шару. Вертикальні архітектури, у свою чергу, можна поділити на *однопрохідні* (коли до входу підключений самий нижній шар, а до виходу самий верхній) та *двопрохідні* (коли і до входу, і до виходу підключений нижній архітектурний шар).

Різні типи архітектур схематично зображено на рис. 1.

Великою перевагою багатошарових архітектур є модульність, а також можливість поєднання різних способів реалізації. Якщо агент повинен демонструвати n різних типів поведінки, то ми можемо ввести n архітектурних шарів і реалізувати кожен тип поведінки найбільш зручним для цього способом.

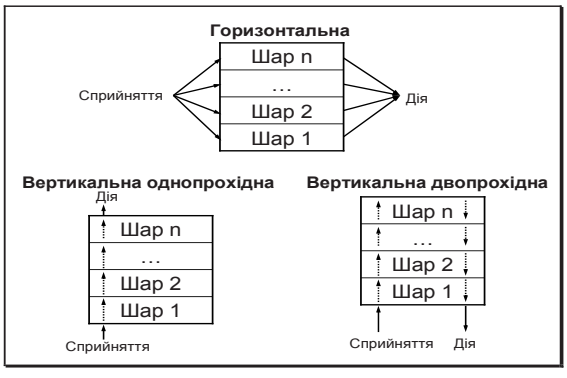


Рис. 1. Типи багатошарових архітектур реалізації агента

Багатошарова архітектура породжує проблему узгодження рішень, прийнятих на різних архітектурних шарах. Адже те, що кожен з шарів окремо виконує своє завдання коректно, не гарантує того, що об'єднання відповідних рішень також буде коректним.

У горизонтальних архітектурах для усунення проблеми узгодження можна, зокрема, використовувати єдиний модуль-диспетчер, що приймає рішення про те, який саме з архітектурних шарів повинен управляти агентом в даний момент. Але такий диспетчер може значно обмежити ефективність системи з двох причин. По-перше, дуже непросто такий механізм реалізувати. Наприклад, в архітектурі передбачено n шарів, кожен з яких здатен запропонувати m можливих дій, отже, розробнику контролюючого механізму доведеться врахувати m^n можливих сценаріїв взаємодії шарів, що може мати досить високу обчислювальну складність. По-друге, такий централізований механізм є вузьким місцем системи, що часто виявляється неефективним і ненадійним.

Зазначена проблема частково усувається у вертикальних архітектурах. Розглянемо, наприклад, двопрохідну вертикальну архітектуру, в якій вхідні дані спочатку піднімаються з нижнього шару до верхнього, а потім прийняте рішення спускається з верхнього шару до нижнього. У разі n архітектурних шарів, кожен з яких може запропонувати m дій, ми маємо 2*m*(n - 1) можливих сценаріїв взаємодії, причому їх реалізація не зосереджена в одному місці, а розподілена по шарах. Однак вертикальні архітектури мають свої серйозні недоліки. Наприклад, управління повинне пройти через кожен архітектурний шар, перш ніж рішення буде ухвалено.

Це, по-перше, не надто ефективно, а по-друге, небезпечно – збій в будь-якому з шарів призводить до повного краху системи.

Для нашої задачі будемо використовувати тришарову вертикальну однопрохідну архітектуру, де кожен шар буде виконувати одну з підзадач.

3. Вибір оптимального рішення агентом

Агенту на кожному кроці слід вибрати найкращу дію, виходячи з того, що йому відомо про навколишній світ.

В ході дослідження даної проблеми було розглянуто декілька методів для її вирішення. Порівняль-

на характеристика розглянутих методів приведена в табл. 1.

Таблиця 1

Порівняльна характеристика методів прийняття оптимального рішення

Метод	Переваги	Недоліки
Метод ітеративного виключення строго домінуючих дій	• Підходить для функціонування в одноагентному середовищі. • Допускається використання різних стратегій різними агентами	• Потребує модель переходу. • У випадку мультиагентного середовища потребує припущення в загальних знаннях, що всі агенти є раціональними.
Метод на основі рівноваги Неша	• Породжує більш точні передбачення результатів в широкому класі ігор.	• Не підходить для одноагентних середовищ.
Алгоритм знаходження оптимальної дій Парето	• Передбачена змішана стратегія агента (можлива поява стохастичності в поведінці агента).	• Розрахований лише на спільну дію агентів.

Серед приведених методів для вирішення поставленої задачі було обрано метод ітеративного виключення строго домінуючих дій, який було адаптовано під конкретні завдання.

Нехай на кожному кроці t = 1, 2, ..., ∞ сприйняття середовища агентом можна записати як o_t, а із скінченої множини дій A вибрати одну дію a_t, тоді в загальному випадку функція $f(o_1,a_1,o_2,a_2,...,o_t)=a_t$ буде стратегією агента, де a_t – оптимальна дія.

Якщо агент моделюється для середовища з повним спостереженням, у більшості випадків стратегія приймає вигляд $f(o_t)=a_t$, тобто ігнорується вся історія спостережень окрім останнього. В нашому ж випадку задача поставлена для середовища з частковим спостереженням. Для нього o_t описує лише частину інформації про стан системи, що задає розподілення ймовірностей $P(s_t \mid o_t)$ між дійсними станами системи (s_t – стан системи в момент часу t) [9].

Після вибору дії агентом світ змінюється як його наслідок. Модель переходу (модель світу) визначає як змінюється світ після виконання дії. Якщо в окремий момент часу стан системи має значення s_t й агент виконує дію a_t, можна виділити два випадки:

• в *детермінованому* середовищі модель переходу має вигляд $(s_t,a_t) \rightarrow s_{t+1}$;

• в *стохастичному* середовищі модель переходу відображає пару стан-дія у вигляді розподілення ймовірностей $(s_t,a_t) \rightarrow P(s_{t+1} \mid s_t, a_t)$, тобто ефект дії агента не відомий наперед.

Очевидно, що стохастичність при переході з одного стану в інший вносить додаткові складності в задачі прийняття оптимального рішення агентом.

Введемо поняття *корисності стану для агента*, яке має вигляд U(s) для s стану середовища, таке що для двох станів s та s* виконується U(s) > U(s*), тоді й тільки тоді, коли для агента вигідніший стан s ніж стан s* середовища, а U(s) = U(s*) у тому випадку, коли для агента ці стани однакові.

Тепер, маючи поняття користності стану, можна вивести функцію знаходження оптимальної дії. Прийняття рішення основане на припущені, що дія агента має бути максимумом корисності стану. Звідси маємо:

$$a_t^* = \arg\max U(s_{t+1}) \quad (1)$$

для детермінованого середовища, та

$$a_t^* = \arg\max \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) U(s_{t+1}) \quad (2)$$

для стохастичного [10]. Тобто, щоб дізнатися наскільки корисна дія, агент повинен знайти добуток корисності кожного можливого стану та ймовірності потрапляння в цей стан, а потім підсумувати одержаний результат. Тоді агент зможе вибрати дію a_t^*, у якої буде максимальна сума.

В більшості випадків в середовищі агент співпрацює з іншими агентами для досягнення спільної цілі, при цьому вони можуть мати різні виконуючі механізми та підзідачі. В такому разі використовуючи (1) та (2) є ймовірність отримати підоптимальне рішення. Під підоптимальним рішенням мається на увазі дія, котра незалежно від того, що роблять інші агенти, завжди буде в результаті менш виграшною для агента, ніж яка-небудь інша дія. В цьому випадку *алгоритм ітеративного виключення строго домінуючих дій* запишеться наступним чином: дія a_i агента і є строго домінуюча над іншими діями діями a_t', якщо:

$$u_i(a_{-i},a_i') > u_i(a_{-i},a_i) \quad (3)$$

для всіх дій a_{-i} інших агентів. В цьому визначенні $u_i(a_{-i},a_i)$ являється виграшем для агента і, котрий він отримує, якщо вибере дію a_i в той час як інші агенти виконують дії a_{-i}.

4. Підтвердження або спростування гіпотези можливої дії

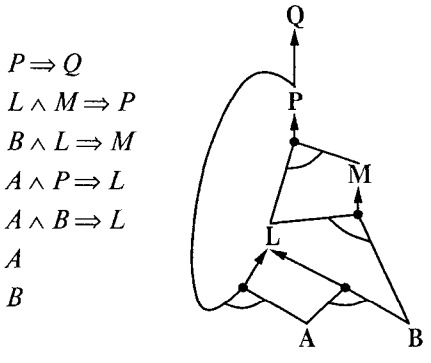
Критерієм вибору методу вирішення даної підзадачі були рішення про створення агента на основі пропозиційної логіки та простота в реалізації алгоритму. З порівняльної табл. 2 видно, що даним критеріям відповідає алгоритм зворотнього логічного виведення [2].

Таблиця 2

Порівняльна характеристика методів пошуку рішення

Метод	Переваги	Недоліки
Алгоритм прямого логічного виведення	• Призначений для агентів, які основані на пропозиційній логіці. • Простий в реалізації. • Являється несуперечливим. • Виконує роботу за лінійний час.	• Час виконання залежить від розміру бази знань агента. • Важчий в реалізації для логіки високих порядків.
Алгоритм зворотнього логічного виведення	• Має вісі переваги алгоритма прямого логічного виведення, проте швидший за нього та зменшує наповнення бази знань надлишковою інформацією	• Час виконання залежить від величини бази знань агента. • Важчий в реалізації для логіки високих порядків
алгоритм пошуку з поверненням (Девіса-Патнем)	• Є швидшим за два попередні.	• Досить складний в реалізації. • Потребує великого набору логічних правил представлення інформації.

Продемонструємо цей алгоритм на прикладі. На рис. 2(а) показана проста база знань з хорновськими виразами, що містить вирази A і B як відомі факти. На рис. 2(б) приведена та ж база знань, зображена у вигляді графа І-АБО. У графах І-АБО кратні дуги, з'єднані кривою лінією, позначають кон'юнкцію (в них необхідно довести істинність кожної дуги), а кратні дуги, не сполучені один з одним, позначають диз'юнкція (досить довести істинність будь-який з цих дуг).



а) б)

Рис. 2. Приклад простої бази знань у вигляді хорновських фраз (а) та у вигляді І-АБО графу (б)

Якщо вдається відразу ж дізнатися, що вислів, який міститься в запиті Q, є дійсним, то не потрібно виконувати ніякої роботи. В іншому випадку алгоритм знаходить ті імплікації в базі знань, з яких випливає Q. Якщо можна довести, що всі передумови однієї з цих імплікацій є істинними (за допомогою зворотного логічного виведення), то вислів Q також є істинним. Застосувавши до запиту Q, що зображений на рисунку, цей алгоритм буде проходити вниз по графу до тих пір, поки не досягне множини відомих фактів, які утворюють основу для доказу.

Зворотнє логічне виведення являє собою одну з *форм міркування, що направляється цілями*. Така форма є корисною при отриманні відповідей на конкретні запитання агента. Найчастіше витрати на зворотнє логічне виведення набагато менші в порівнянні з вартістю, лінійно залежної від розміру бази знань, оскільки в цьому процесі задіяні тільки факти, які безпосередньо відносяться до справи.

5. Самонавчання агента

Аналогічно підходу вирішення попередніх двох підзадач для проблеми самонавчання агента було розглянуто декілька методів (табл. 3). Додатково з критеріями вибору, що згадуються в попередньому розділі, при виборі алгоритму бралася до уваги відсутність початкової вибірки фактів. Таким чином, для реалізації даної підзадачі використовується алгоритм Q-Learning.

Розглянемо цей алгоритм для випадку, коли наше середовище стохастичне. Припустимо, що для кожного стану зовнішнього середовища визначена деяка функція R(s), яка визначає корисність стану s для нашого агента. Агенту потрібно знайти таку послідовність дій, що в підсумку він прийде до максимально корисного стану.

Q-Learning алгоритм вираховує величини Q(a,s) (корисність дії a в стані s середовища). Ці величини на початку мають випадкове значення, а потім сходяться з деяким значенням в ході ітерацій присвоєння після кожної дії агента в процесі навчання.

$$Q(s,a) = (1-\lambda)Q(s,a) + \lambda(R(s) + \gamma \max Q(s',a')), \qquad (4)$$

де $\lambda \in (0,1)$ – доля, на яку ми дозволяємо кожному присвоєнню змінювати значення Q(a,s),

$\gamma \in [0,1)$ – коефіцієнт зменшення, який використовується для того, щоб величина Q зійшлася до якогось сталого значення.

Таблиця 3

Порівняльна характеристика алгоритмів реалізації самонавчання

Метод	Переваги	Недоліки
Алгоритм Q-Learning	• Простий в реалізації. • Не потребує початкової вибірки фактів.	• Потребує вдалого вибору коефіцієнтів λ та γ для швидшого визначення функції навчання.
Алгоритм пошуку найкращої поточної гіпотези	• Добре працює для пропозиційної логіки та логіки першого порядку.	• Передбачає супровід єдиної гіпотези та її коректування по мірі додаваня нових прикладів. • Потребує великої кількості прикладів навчальної виборки.
Алгоритм індуктивного навчання на основі знань	• Дає кращі результати в порівнянні з попередніми. Добре зарекомендував себе в реалізації нейронних мереж.	• Досить складний в реалізації. • Потребує навчальної вибірки фактів.

6. Реалізація програмного комплексу

Для програмної реалізації тестових задач використано мову об'єктно-орієнтованого програмування C++ та середовище розробки Borland C++ Builder 6. Загальна схема структури програмного продукту зображена на рис. 3.

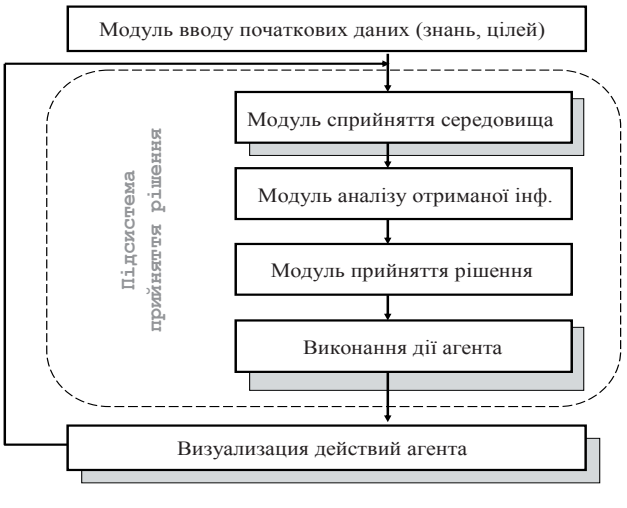


Рис. 3. Структура програмного продукту

На початку роботи інтелектуального агента йому задаються початкові знання про середовище, такі як карта світу, ймовірні перепони. Також на цьому етапі для агента встановлюються цілі, яких йому необхідно досягти.

В ході роботи агент, за допомогою своїх датчиків чи сенсорів, сприймає деяку інформацію про середовище та аналізує її.

Для агента-робота даний модуль представлений у вигляді окремого програмного комплексу, який перетворює сигнали та зображення від сенсорів у більш зрозумілу для агента-програми форму. Отримана інформація приводиться до пропозиційної логіки та потрапляє до бази знань.

Модуль прийняття рішення представлений у вигляді тришарової вертикальної однопрохідної системи, де кожен з шарів використовуючи базу знань агента вирішує одну з підзадач описану вище.

В ході програмної реалізації підзадачі підтвердження/спростування гіпотези про можливу дію агента база знань мала тенденцію динамічно розростатися. Це призводило до швидкого вичерпування ресурсів пам'яті.

Даний недолік було усунено кешуванням проміжних даних виведення та подальшим усуненням надлишової інформації серед кешованих даних за рахунок використання правил резолюції та факторизації.

Звідси видно, що база знань агента має статично-динамічну структуру. Вона має незмінні початкові знання про середовище, від яких відштовхується при прийнятті рішення, факти-знання, що надходять до агента у якості сприйняття середовища (стани середовища), а також висловлювання, що додаються до бази знань після прийняття рішення агентом.

Останнім модулем є модуль виконання дії агента. Для агента-робота даний модуль представлений у вигляді окремого програмного комплексу, котрий на вхід отримує чергу дій зі списку можливих дій, які може виконати робот, перетворює цю інформацію на команди та взаємодіє з рельними виконуючими механізмами робота.

7. Визначення тестових задач

Для реалізації математичної моделі, що описана вище, та її тестування було створено програмний комплекс для таких задач:

- виживання агента в класичному світ вампуса з модифікаціями;
- бортова система робота-розвідника, що діє в радіоактивному приміщенні для знаходження та збору радіоактивних часток.

Як видно з табл. 4, ці задачі мають багато спільних характеритик.

Таблиця 4

Опис тестових задач

	Агент, що виживає у світі вампуса	Бортова система робота-розвідника, що діє в радіоактивному приміщенні
Аналогія середовища та агентів	• карта світу вампуса	• карта приміщення
	• статичні та динамічні першкоди	• статичні перешкоди
	• набір можливих дій агента	• набір можливих рухів робота
	• набір датчиків сприйняття інформації	• набір сенсорів робота
	• цілі – знайти золото, вбити вампуса, повернутися в початкове місцезнаходження	• цілі – знайти радіоактивні частинки, встигнути повернутися на базу допоки є заряд елементу живлення

8. Експериментально отримані результати

Для задачі виживання агента у світі вампуса було створено окремий програмний комплекс, що порівнює за деякими параметрами модифіковані алгоритми та підходи до реалізації поставлених задач з базовими.

На даних прикладах відображається тестування вдосконалених алгоритмів з використанням методу ітеративного виключення строго домінуючих дій та зворотнього логічного виведення в порівнянні з дією алгоритмів-аналогів без додаткових методів раціоналізації. Агенту було поставлено ряд цілей, які він мав виконати.

При виконанні завдання оптимальність його дій вираховувалася наступним чином:

- за досягнення будь-якої цілі агенту до загального рахунку додаються бали як заохочення;

Рис. 4. Приклади тестування модифікованих алгоритмів

• за кожну дію, яку виконує агент, з загального рахунку знімається певна кількість балів;

• загальна сума на момент повернення в початкове місцезнаходження є його коефіцієнтом оптимальності для даної карти.

Таким чином, порівняння агентів реалізованих різним чином проводилося за такими параметрами:

• час, витрачений на досягнення цілей (в тіках);

• максимальна кількість пам'яті, що була використана (в байтах);

• кількість здобутих балів.

На рис. 4 наведеноотримані результати для порівняння.

Як видно з таблиці характеристик на рисуку 4, інтелектуальний агент на основі вдосконаленх методів, досягає поставлених цілей в 2–3 рази швидше. При цьому він використовує в 3–3,5 разів менше комп'ютерної пам'яті та є в 1,5 рази оптимальнішим.

Висновки

1. Проаналізовано та виділено характеристики середовища та агента для даного середовища.

2. Обгрунтовано побудову архітектури агента на основі багатошарових агентів.

3. Обгрунтовано вибір методів реалізації поставлених задач для створення інтелектуального агента, та викладено математичну модель цих методів адаптовану під визначені властивості агента і середовища.

4. Підтверджено математичну модель експериментальними даними.

Література

1. Vidal, Jos'e M. Fundamentals of Multiagent Systems/ Jos'e M. Vidal. – 2007. – 157 p.
2. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд. - М.: Издательский дом «Вильямс», 2006. – 1408 с.
3. R. H. Bordini, J. F. H bner, M. Wooldridge. Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley and Sons Ltd, October 2007. – 292 p.
4. W. van der Hoek and M. Wooldridge. Multi-Agent Systems// F. van Harmelen, V. Lifschitz, and B. Porter, editors, Handbook of Knowledge Representation, pages 887 – 928. Elsevier, 2008. – Режим доступу: http://www.csc.liv.ac.uk/~mjw/pubs/kr-handbook.pdf
5. N. Troqard, W. van der Hoek, and M. Wooldridge. A logic of propositional control for truthful implementations.// Theoretical Aspects of Rationality and Knowledge (TARK XII). Stanford University, CA, July 2009. - Режим доступу: http://www.csc.liv.ac.uk/~mjw/pubs/tark2009.pdf
6. D. Peled and M. Wooldridge. Model Checking and Artificial Intelligence (LNCS Volume 5348). Springer-Verlag, March 2009. - Режим доступу: http://www.csc.liv.ac.uk/~mjw/pubs
7. Бобровский С. Искусственная жизнь и автономные агенты. - Режим доступу: http://www.ai.obrazec.ru/ai_program.htm
8. Кузьмин. Д. А., Селютин Д. Г., Покидышева Л. И. Анализ средств разработки мультиагентных систем // Вестник Красноярского государственного университета (КрасГУ): Физико-математические науки. – 2006. – №7. - Режим доступу: http://library.krasu.ru/ft/ft/_articles/0112838.pdf.
9. Бугайченко Д. Ю., Соловьев Д. Ю. Абстрактная архитектура интеллектуального агента и методы ее реализации // Системное программирование. Вып. 1. СПб.: Изд-во СПбГУ, 2005. С. 36-67. - Режим доступу: http://www.sysprog.info/2005/03.pdf.
10. Третьяков Д. І., Шаповалова С. І. Моделювання раціонального агента в середовищі зі змінними характеристиками// Сучасні проблеми наукового забезпечення енергетики: Тези доповідей VIII Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів (19-23 квітня 2010 року).– Київ: НТУУ «КПІ», 2010. – с. 354.