ПРИЛАДОБУДУВАННЯ ТА ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА ТЕХНІКА

УДК 681.5.015.24(045)

В.О. Апостолюк

ПОШУКОВІ АЛГОРИТМИ ПРОЦЕДУРНОГО ОПТИМАЛЬНОГО КЕРУВАННЯ

Вступ

Пошукові, або хвильові, алгоритми досить широко використовуються в теорії зв'язаних графів для розв'язання задачі пошуку найкоротшого шляху між двома даними станами серед скінченної множини станів системи [1]. Багато з модифікацій цих алгоритмів можуть також використовуватися в задачах штучного інтелекту або для пошуку в деревоподібних структурах даних [2, 3]. Безпосереднє використання цих алгоритмів для пошуку в суцільному просторі станів, що необхідне для оптимального процедурного керування динамічними системами, є проблематичним через нескінченну кількість станів після дискретизації простору, з одного боку, і зниженням точності знайденого керування, з іншого. Крім того, розв'язання задач оптимального керування з використанням існуючих алгоритмів вважається непрактичним з огляду на експоненціальне зростання кількості станів, які слід аналізувати. У цьому плані вважається актуальною і доцільною розробка нових алгоритмів на їх основі, які могли б успішно розв'язувати задачі процедурного оптимального керування в усіх областях його застосування.

Дана стаття присвячена результатам розробки і дослідження нових алгоритмів, які б могли ефективно використовуватися для розв'язання задач процедурного оптимального керування динамічними системами в реальному часі. Одним із видів алгоритмів, які можуть бути модифіковані для оптимального керування, є так звані хвильові алгоритми.

Постановка задачі

Загальна постановка задачі оптимального керування є достатньо відомою [4]. Потрібно лише нагадати про деякі аспекти, важливі для подальшого дослідження. Нехай стан системи в момент часу t задається вектором $\mathbf{x}(t) = \{x_1(t), ..., x_n(t)\}$ в n-вимірному евклідовому просторі станів X. Керуючі впливи на об'єкт керування задаються m-вимірною векторною функцією часу $\mathbf{u}(t) = \{u_1(t), ..., u_m(t)\}$. Компоненти вектора керування $\mathbf{u}(t)$ є кусково-сталими і обмеженими таким чином, що у будь-який момент часу t вони знаходяться в деякій обмеженій області U простору керуючих впливів. Не зменшуючи загальності постановки задачі, можна вважати, що $|u_i| \le 1$, $i = 1, ..., m$. Такі керуючі впливи є прийнятними з точки зору досліджуваних алгоритмів.

Будемо вивчати системи, поведінку яких можна промоделювати за допомогою найбільш загальної функції зміни стану F, що дозволяє прогнозувати стан системи через проміжок часу Δt на основі стану системи в момент часу t і прикладеного керування $\mathbf{u}(t)$:

$$\mathbf{x}(t + \Delta t) = F(\mathbf{x}(t), \mathbf{u}(t)) . \qquad (1)$$

Функція F у формулі (1) вважається визначеною для всіх $\mathbf{x} \in X$ і $\mathbf{u} \in U$. Такий підхід до задання об'єкта керування значно розширює класи систем, які можуть керуватися за допомогою розроблених алгоритмів, порівняно з класичними методами оптимального керування, коли система задається системами диференціальних рівнянь.

Тепер потрібно знайти таке керування для системи (1), яке переведе її з початкового стану $\mathbf{x}_0 = \mathbf{x}(t_0)$ у відомий очікуваний кінцевий стан $\mathbf{x}_g = \mathbf{x}(t_g)$ і мінімізує або максимізує деяку задану функцію якості. Таку функцію можна задати, наприклад, у формі деякого функціонала якості:

$$J(\mathbf{x}_0 \to \mathbf{x}_g) = \int_{t_0}^{t_g} f_0(\mathbf{x}(t), \mathbf{u}(t))dt . \qquad (2)$$

У даному випадку, звісно, вважатимемо, що функції керування, які переводять систему із стану $\mathbf{x}_0$ у стан $\mathbf{x}_g$, взагалі існують, і серед них слід шукати такі, які мінімізують функцію якості J. Хоч функція (2) задана в традиційній формі функціонала, однак це не є необхідною вимогою для досліджуваних алгоритмів. Наразі будь-яка форма задання критерію якості має бути прийнятною для алгоритмів процедурного оптимального керування.

Процедурне оптимальне керування

Традиційні підходи до оптимального керування за допомогою алгоритмів базуються на результатах аналітичного або чисельного аналізу варіаційної задачі відносно вибраного критерію якості і обмежень у системі. На відміну від цього процедурне оптимальне керування будується навколо спеціального алгоритму, який використовує модель об'єкта керування виключно для

прогнозування його стану за умов обмежень і функції керування, і ні в якому разі не залежить від самого об'єкта або обмежень у системі. У цьому розумінні процедурне керування є формою прямого керування з прогнозуючою математичною моделлю, яка також забезпечує оптимальність даного керування відносно вибраного критерію якості. Такий підхід дає можливість розробляти універсальні обчислювальні системи оптимального керування, які зовсім не потребують змін в апаратному забезпеченні для застосування їх як керуючих пристроїв для будь-яких об'єктів. Власне кажучи, єдиним компонентом такої системи, який потребує адаптації до конкретного об'єкта керування, є його математична модель та зумовлені ціллю керування критерій якості і обмеження.

Із врахуванням складності моделі об'єкта керування і складності форми подання обмежень можливі області застосування процедурного оптимального керування порівняно з іншими методами наведені на рис. 1. Чим більша ефективність розв'язання задачі оптимального керування в наведених на рисунку методах, тим темнішим кольором їх виділено. Треба також зазначити, що класичні методи і нечіткі системи не дають оптимальних, з точки зору вибраного критерію якості, розв'язків задачі керування.

Щодо зазначених областей застосування, то такі алгоритми повинні мати спеціальні властивості, які б дозволяли їм ефективніше розв'язувати задачі оптимального керування порівняно з існуючими методами:

- математична модель об'єкта керування має використовуватись виключно для прогнозування стану об'єкта від керуючих впливів. Ніяких додаткових обмежень щодо лінійності моделі або її простоти не повинно вводитись;
- керуючий алгоритм має ефективно адаптуватися до змін параметрів прогнозуючої моделі;

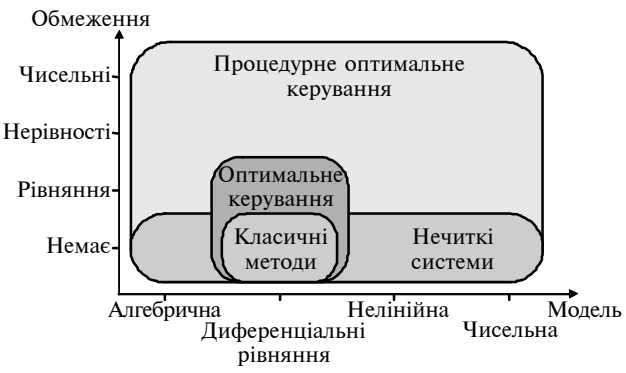

Рис. 1. Області застосування процедурного оптимального керування

- безпосередньо мають враховуватись критерій оптимальності і обмеження, які можуть подаватися в довільній формі.

Остання властивість дає змогу таким алгоритмам розв'язувати, наприклад, задачу формування оптимальної траєкторії руху за наявності перешкод та із врахуванням власної динаміки об'єкта і одночасним формуванням функції керування, яка б реалізувала рух об'єкта по цій траєкторії.

Узагальнену схему функціонування алгоритму як частину загальної системи керування наведено на рис. 2. На ній $\mathbf{x}_g$ – цільовий стан системи, $\mathbf{x}$ – стан, який прогнозується за допомогою математичної моделі, $\mathbf{x}^*$ – реальний стан системи, $\mathbf{u}$ і $\mathbf{u}^*$ – керування, яке аналізується, та знайдене оптимальне керування, відповідно. Похибка прогнозування стану об'єкта керування використовується для модифікації моделі системи з точки зору покращення її прогнозів. Алгоритм використовує модель об'єкта для прогнозування стану системи в процесі пошуку найкращого керування, яке має привести систему до бажаного стану $\mathbf{x}_g$. Як тільки таке оптимальне керування $\mathbf{u}^*$ буде знайдено, воно подається на об'єкт. Для деяких видів алгоритмів процедурного керування процес пошуку не закінчується після знаходження оптимального керування, а продовжується з метою можливого покращення здобутого раніше результату, особливо в разі модифікації моделі об'єкта керування.

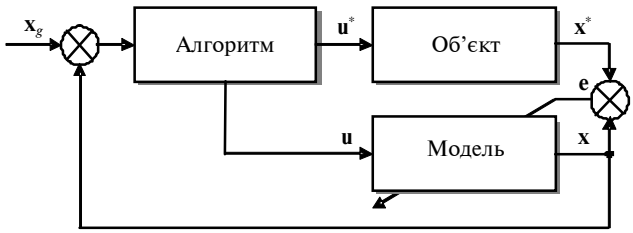

Рис. 2. Схема функціонування алгоритму

При функціонуванні систем процедурного керування можна виділити такі три режими їх роботи:

- пошук функції керування з використанням моделі системи. При цьому керуючі впливи на об'єкт керування не формуються;
- застосування знайденої функції для керування об'єктом, верифікації моделі та можливі її покращення;
- модифікація функції керування, якщо результати керування незадовільні.

Потреба в останньому режимі, а саме модифікації знайденої функції керування, може ви-

никнути через різні причини, в тому числі через модифікацію моделі, яка може виконуватися в іншому режимі. Для деяких алгоритмів модифікація функції керування означає, що процес пошуку оптимального керування необхідно починати з самого початку, що означає перехід системи до першого режиму. Очевидно, що алгоритми, які не потребують цього, є найпривабливішими з точки зору процедурного керування. Також треба зазначити, що у загальному випадку абсолютно різні методи або їх ефективна комбінація можуть використовуватися для реалізації різних режимів роботи системи.

Задання функції керування

Для того щоб сформулювати алгоритм розв'язання задачі оптимального керування, спочатку розглянемо форму задання функції керування, яку необхідно знайти. Будемо вважати, що кожний компонент вектора керування $\mathbf{u}(t) = \{u_1(t), ..., u_m(t)\}$ є функцією часу, але протягом проміжку часу Δt може набувати значень тільки з фіксованої множини сталих величин:

$$\mathbf{u}^0 = \{u_1^0, ..., u_p^0\}. \qquad (3)$$

У цьому випадку існує $s = p^m$ можливих сталих значень вектора керування $\mathbf{u}(t) \in \mathbf{u}_1^0, ..., \mathbf{u}_s^0$. Подача цих кусково-сталих функцій керування на систему в стані $\mathbf{x}_i = \mathbf{x}(t_i)$ може призводити до s нових станів системи через кожний проміжок часу Δt. Очевидно, що таке експоненціальне зростання кількості станів системи, яке потрібно проаналізувати, потребує спеціальних алгоритмів, здатних вирішувати дану проблему в реальному часі. Щодо розв'язання початкової проблеми оптимального керування, ми шукаємо таку скінченну послідовність з v керуючих впливів $\mathbf{u}_i(t)$, які переводять систему із стану $\mathbf{x}_0$ в цільовий стан $\mathbf{x}_g$. Таку послідовність називатимемо *керуючою програмою*

$$B = \{b_1, ..., b_v\},\ b_i \in [1, ..., p], \qquad (4)$$

де кожний компонент, або *керуюча команда*, являє собою індекс сталого керування з вектора можливих значень (3). У випадку багатовимірного керування ($m > 1$) команди для кожного компонента вектора керування $\mathbf{u}(t) = \{u_1(t), ..., u_m(t)\}$ беруться з програми (4) послідовно. В разі, якщо кількість команд у керуючій програмі v не є кратною розмірності вектора керування m, то необхідні команди беруться такими, що дорівнюють деякій наперед визначеній команді.

В результаті поведінка об'єкта керування визначається дією на нього кусково-сталої векторної функції керування, компоненти якої мають такий вигляд:

$$u_i(k\Delta t) = u_{b_k}^0,\ k \in [1, ..., v]. \qquad (5)$$

Тут кожний сталий керуючий вектор $\mathbf{u}_i$ діє на систему протягом наперед заданого інтервалу часу Δt. Графічно цю функцію зображено на рис. 3.

Треба зазначити, що хоч кусково-стале керування є найпростішою формою зображення функції керування, інші форми інтерпретації команд керуючої програми, в тому числі логічні булевські команди, також не заперечують принципам функціонування розроблених алгоритмів.

Крім того, факт дискретизації простору станів на основі вибраного кроку Δt призводить до того, що знайдене керування не є оптимальним у глобальному розумінні, але нескінченно наближається до нього при $\Delta t \to 0$.

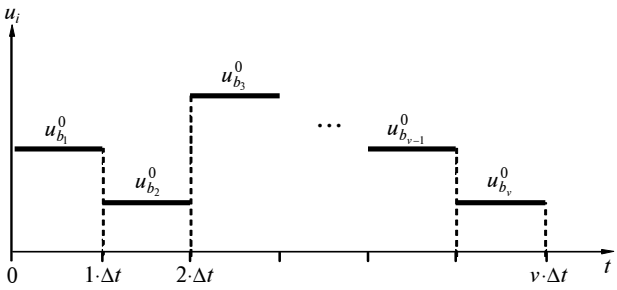

Рис. 3. Кусково-стала функція керування

Узагальнена структура пошукових алгоритмів

Більшість пошукових, або хвильових, алгоритмів можна описати за допомогою узагальненої структури, наведеної на лістингу в табл. 1 [5].

Наприклад, відомі алгоритми пошуку в ширину, Дайкстри [1] і A-Star [2, 3] будуть різнитися лише змістом підкреслених у лістингу функцій "Equal", "Constrained", "Cost" і "Score". Ці чотири функції, а також конкретний тип таких базових змінних, як пріоритетна черга *Open* і список *Closed* повністю визначають роботу будь-якого алгоритму, розробленого за даною структурою.

У наведеній структурі алгоритму вважається, що змінні векторів станів можуть також зберігати значення функції якості (*Cost*) в цьому стані, величину рейтингу цього стану (*Score*), а також вектор керування, який спричинив перехід системи до цього стану. Нові, "нащадкові", стани об'єкта керування $\mathbf{x}_{n+1}$ розраховуються за допомогою моделі системи (1) із стану $\mathbf{x}_n$ прикла-

денням до об'єкта всіх можливих значень вектора керувань, які визначаються розмірністю керування m та розмірністю множини (3). В разі, коли хоча б один стан системи $\mathbf{x}_n$ збігався в значенні функції *Equal* із цільовим станом $\mathbf{x}_g$, алгоритм успішно завершує свою роботу (крок 4), після чого проводиться зворотна реконструкція послідовності керуючих векторів, які привели систему до цільового стану. Важливою особливістю таких алгоритмів є використання саме пріоритетної черги станів *Open*, що будується на основі рейтингу та за умови правильного вибору рейтингової функції *Score* здатна значно зменшити кількість станів, які необхідно проаналізувати для знаходження найкращого керування. Крім того, якщо рейтинг кожного стану відповідає значенню критерію якості в цьому стані, то знайдена послідовність керуючих впливів буде також оптимальною з точки зору вибраного критерію.

***Таблиця* 1.** Структура пошукового алгоритму

Крок	Алгоритм
1	push $\mathbf{x}_0$ on *Open*
2	**while** *Open* is not empty **do**
3	pop $\mathbf{x}_n$ from *Open*
4	**if** Equal($\mathbf{x}_n$, $\mathbf{x}_g$) **then return** Success
5	**for** each successor $\mathbf{x}_{n+1}$ of $\mathbf{x}_n$ **do**
6	**if** Constrained($\mathbf{x}_{n+1}$) **then continue**
7	$NewCost := \mathbf{x}_n \cdot Cost +$ Cost($\mathbf{x}_n$, $\mathbf{x}_{n+1}$)
8	**if** ($\mathbf{x}_{n+1}$ is in *Open* or *Closed*) **and** ($\mathbf{x}_{n+1} \cdot Cost <= NewCost$) **then continue**
9	$\mathbf{x}_{n+1} \cdot Cost := NewCost$
10	$\mathbf{x}_{n+1} \cdot Score :=$ Score($\mathbf{x}_{n+1}$)
11	**if** $\mathbf{x}_{n+1}$ is in *Closed* **then** remove $\mathbf{x}_{n+1}$ from *Closed*
12	**if** $\mathbf{x}_{n+1}$ is not in *Open* **then** push $\mathbf{x}_{n+1}$ on *Open*
13	push $\mathbf{x}_n$ on *Closed*
14	**return** Failure

Спеціальні функції алгоритмів

Як зазначено вище, чотири базові функції, а саме *Equal*, *Constrained*, *Cost* і *Score* повністю визначають принципи роботи алгоритмів, побудованих за наведеною структурою. Крім того, важливим параметром роботи алгоритму процедурного керування є крок моделювання в часі Δt, який може бути як сталою величиною і вибиратися заздалегідь, так і змінною, яка адаптується в процесі роботи алгоритму.

Розглянемо зміст цих базових функцій алгоритмів детальніше.

Функція *Equal* є функцією булевського типу, яка порівнює два стани об'єкта керування і повертає значення "істина" (True), якщо ці стани можна вважати однаковими за вибраним критерієм. Найпростішою реалізацією даної функції може бути порівняння евклідової відстані в просторі станів між цими двома станами із заданою пороговою функцією

$$\left|\mathbf{x}_i - \mathbf{x}_j\right| \le D(\mathbf{x}_i, \mathbf{x}_j). \qquad (6)$$

Неважко побачити, що порогова функція $D(\mathbf{x}_i, \mathbf{x}_j)$ є також похибкою, з якою система буде переведена до цільового стану. Тому в багатьох випадках її можна вибрати сталою і пропорційною до прийнятної похибки керування. Треба зазначити, що ця функція також використовується в процесі пошуку необхідного стану в черзі *Open* і списку *Closed*.

Функція *Constrained* є також булевською функцією, яка повертає значення "істина" в тому випадку, коли стан $\mathbf{x}_{n+1}$ забороняється за вибраним критерієм або підпадає під задані обмеження. Треба сказати, що з погляду на роботу алгоритмів як конкретна реалізація цієї функції, так і спосіб задання обмежень є абсолютно неістотними.

Функція якості, або цінова функція *Cost*, розраховує величину вибраного критерію якості, наприклад (2), що відповідає переходу системи із стану $\mathbf{x}_n$ до стану $\mathbf{x}_{n+1}$.

Нарешті, функція рейтингу *Score* має забезпечувати спеціальну пріоритезацію станів у черзі *Open*. У класичному алгоритмі A-Star ця функція дорівнює сумі цінової функції $C(\mathbf{x}_0, \mathbf{x}_n)$ для переходу з початкового стану $\mathbf{x}_0$ до стану $\mathbf{x}_n$ та евристичної оцінки $H(\mathbf{x}_n, \mathbf{x}_g)$ переходу із стану $\mathbf{x}_n$ до цільового стану $\mathbf{x}_g$:

$$S(\mathbf{x}_n) = C(\mathbf{x}_0, \mathbf{x}_n) + H(\mathbf{x}_n, \mathbf{x}_g). \qquad (7)$$

Для розрахунку евристичної оцінки $H(\mathbf{x}_n, \mathbf{x}_g)$ може використовуватися евклідова відстань між цими двома станами:

$$H(\mathbf{x}_n, \mathbf{x}_g) = \left|\mathbf{x}_g - \mathbf{x}_n\right|.$$

У подальшому рейтингова функція (7) з такою ж оцінкою буде використовуватися для порівняння з новими функціями, які є кращими з погляду процедурного керування. Однією з таких рейтингових функцій може бути кут між напрямком переходу від попереднього стану до да-

ного та напрямком від попереднього стану до цільового:

$$A(\mathbf{x}_n) = 1 - \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})(\mathbf{x}_g - \mathbf{x}_{n-1})}{|\mathbf{x}_n - \mathbf{x}_{n-1}||\mathbf{x}_g - \mathbf{x}_{n-1}|}. \qquad (8)$$

Рейтингові функції (7) і (8) можна використовувати не тільки окремо, а й в різних комбінаціях:

$$S^*(\mathbf{x}_i) = S(\mathbf{x}_i)\, A(\vec{\mathbf{x}}_i), \qquad (9)$$

$$S^+(\mathbf{x}_i) = w_s\, S(\mathbf{x}_i) + w_a\, A(\mathbf{x}_i). \qquad (10)$$

Тут w_s і w_a – вагові коефіцієнти для відповідних рейтингових функцій.

Всі наведені вище рейтингові функції (7)–(10) відповідають різним алгоритмам, властивості яких при розв'язанні задач оптимального керування необхідно дослідити і порівняти за їх ефективністю.

Тестування алгоритмів

Вважається за доцільне проводити тестування алгоритмів на задачі, розв'язок якої добре відомий. Розглянемо лінійну систему, яка задається в просторі станів таким чином:

$$\frac{d}{dt}\begin{bmatrix} X \\ V \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -0{,}5 \end{bmatrix}\begin{bmatrix} X \\ V \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} U. \qquad (11)$$

Таку систему треба перевести з початкового стану $\mathbf{x}_0 = \{1, 1\}$ у кінцевий стан $\mathbf{x}_g = \{0, 0\}$ за умови або мінімальної довжини в просторі станів, що відповідає мінімальному впливу на систему, або за умови мінімального часу керування. Хоч система (11) є достатньо простою і лінійною, але до алгоритмів вона подається виключно у формі чисельної моделі. Крім того, задачу процедурного керування слід розв'язувати за наявності таких обмежень на змінні стану системи:

$$\begin{cases} (X - 1{,}5)^2 + (V - 0{,}5)^2 > 0{,}625, \\ (X - 1)^2 + (V + 0{,}25)^2 > 0{,}625. \end{cases} \qquad (12)$$

Компоненти вектора керування можуть набувати тільки трьох значень $\mathbf{u}^0 = \{-1, 0, 1\}$, які вважаються сталими протягом кроку керування $\Delta t = 0{,}25$ с.

Результати тестування

Оскільки час виконання алгоритмів залежить від потужності обчислювальної машини і найбільша частка загального часу роботи алгоритму втрачається на моделювання динаміки об'єкта керування, то для порівняння ефективності різних алгоритмів пропонується використовувати кількість моделювань системи протягом часу Δt. Одне таке моделювання будемо називати "сім" (скорочено від *simulation*).

Результати порівняльного тестування різних пошукових алгоритмів наведено в табл. 2, а відповідні графіки отриманих функцій керування і фазових траєкторій – на рис. 4–7.

***Таблиця* 2.** Результати тестування алгоритмів

Метод	Стани	"Сіми"	Критерій	Час, мс	Стани	"Сіми"	Критерій	Час, мс
	Мінімальна довжина без обмежень (рис. 4)				Мінімальний час без обмежень (рис. 6)			
S	3890	7455	3,008	6200	3000	3771	3,25	2467
A	**251**	**261**	**3,008**	**48**	304	429	11,25	83
S^*	341	354	3,008	75	353	528	11,25	107
S^+	1683	2217	3,279	939	**1368**	**1572**	**4**	**598**
	Мінімальна довжина з обмеженнями (рис. 5)				Мінімальний час з обмеженнями (рис. 7)			
S	6642	14250	4,48	18331	5694	9177	4,5	10112
A	**366**	**408**	**4,45**	**86**	370	447	10,75	93
S^*	357	393	4,45	83	378	459	10,75	96
S^+	2099	3420	4,48	1453	**3011**	**4275**	**4,75**	**2863**

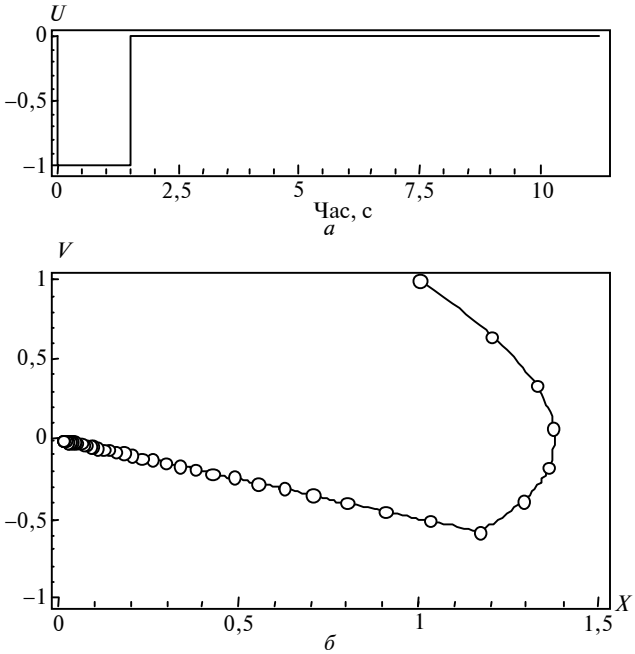

Рис. 4. Мінімальна довжина траєкторії без обмежень: *а* – керування; *б* – фазова траєкторія

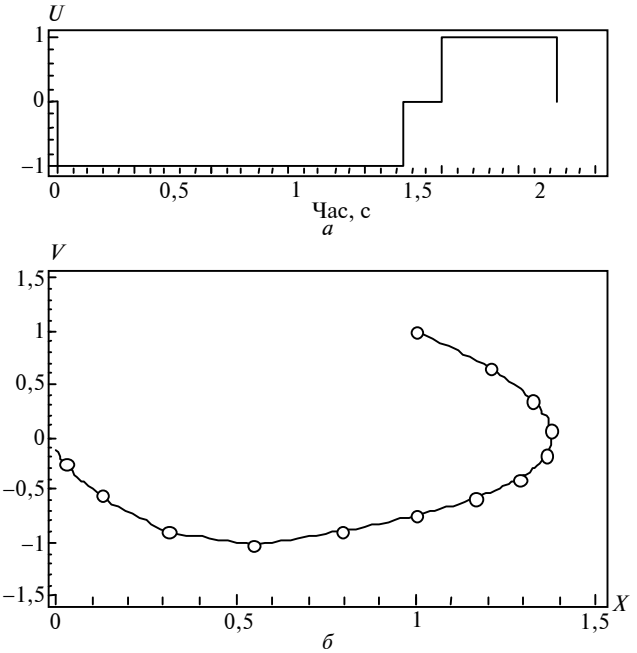

Рис. 6. Мінімальний час без обмежень: *а* – керування; *б* – фазова траєкторія

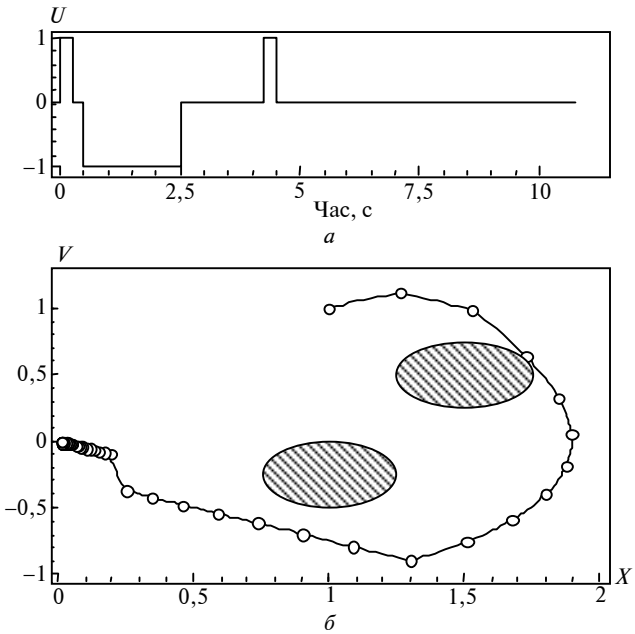

Рис. 5. Мінімальна довжина траєкторії з обмеженнями: *а* – керування; *б* – фазова траєкторія

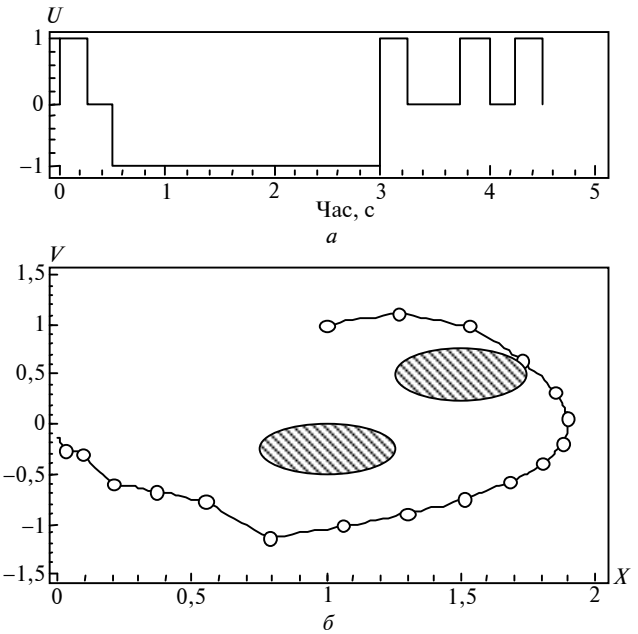

Рис. 7. Мінімальний час з обмеженнями: *а* – керування; *б* – фазова траєкторія

З аналізу наведених результатів можна зробити висновок, що розв'язання задачі керування з мінімальною довжиною фазової траєкторії як з обмеженнями, так і без них отримуємо при використанні рейтингової функції (8). Однак цей алгоритм не найкращий при розв'язанні задачі керування з мінімальним часом, де найкращі результати з погляду на оптимальність отримані за допомогою алгоритму (7). Але зважаючи на низьку ефективність цього алгоритму за кількістю оброблених станів системи і необхідних моделювань, алгоритм (10) є найпривабливіший, оскільки він знаходить майже оптимальне керування при набагато меншій кількості оброблених станів. Треба також зазначити, що отримані результати чітко продемонстрували: для різних критеріїв оптимізації необхідно використовувати різні рейтингові функції.

Новий алгоритм (8) здатний знаходити оптимальне за довжиною фазової траєкторії керування в реальному часі (менше одного такту керування) навіть із використанням не найпотужнішої обчислювальної техніки (Pentium 4 1,7 ГГц, RAM 256 Мб) в кілька десятків разів ефективніший від традиційного A-Star. А втім ефективність роботи пошукових алгоритмів можна значно підвищити за допомогою паралелізації обчислень при моделюванні системи. Прогнозування стану об'єкта керування для різних можливих векторів впливів може виконуватися паралельно. В цьому випадку кількість "сімів", які можуть виконуватись одночасно, становить $s = p^m$, де p – кількість можливих сталих значень компонентів вектора керування, а m – його розмірність.

У публікації [5] було проведено аналіз залежності часу роботи пошукових алгоритмів від кількості оброблених станів. Але для порівняння ефективності пошукових алгоритмів з алгоритмами, в яких окремі стани системи не обробляються, аналіз кількості станів не підходить. На рис. 8 показано залежність часу роботи пошукових алгоритмів від кількості "сімів", яку можна отримати для будь-якого алгоритму, що використовує математичну модель системи.

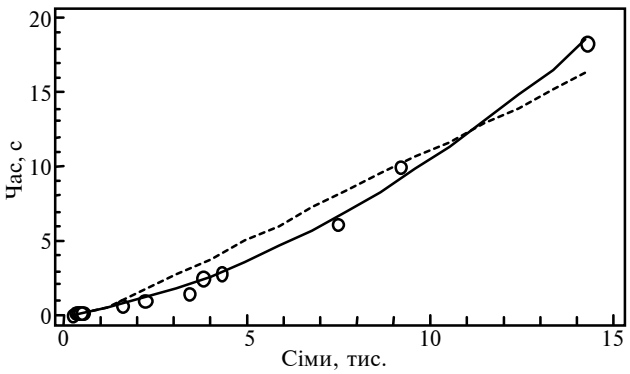

Рис. 8. Залежність часу розв'язання задачі від кількості "сімів": суцільна лінія – квадратична апроксимація; переривчаста – лінійна апроксимація

З аналізу графіків на рис. 8 видно, що залежність часу виконання алгоритму від кількості "сімів" стає практично лінійною для їх великої кількості, коли витрати часу на всі інші обчислення робляться незначними порівняно з витратами на моделювання.

Висновки

З точки зору традиційних підходів до розв'язання задач оптимального керування, запропоновані алгоритми мають такі переваги:

- чисельне, а не аналітичне подання моделі об'єкта керування і обмежень у системі;
- одночасне розв'язання задач побудови оптимальної траєкторії і оптимального керування;
- проста і добре формалізована структура алгоритму, що значно спрощує процес розробки відповідного апаратного забезпечення.

На основі отриманих в даній статті результатів можна зробити висновок, що пошукові алгоритми із спеціальними рейтинговими функціями здатні не тільки знаходити оптимальне керування динамічними системами, а й можуть робити це в реальному часі для окремих простих випадків.

Подальші дослідження повинні проводитися в плані визначення найоптимальніших рейтингових функцій для різних критеріїв оптимізації, а також у напрямку оптимізації параметрів таких функцій.

В.А. Апостолюк

ПОИСКОВЫЕ АЛГОРИТМЫ ПРОЦЕДУРНОГО ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

Рассматривается обобщенная структура поисковых алгоритмов и методы ее адаптации к решению задач оптимального управления многомерными динамическими системами. На основе разработанной структуры предложены и исследованы

V.O. Apostolyuk

SEARCH ALGORITHMS FOR PROCEDURAL OPTIMAL CONTROL

This paper studies a generalized structure of the state traversal algorithms and the methods of their adaptation to procedural optimal control of the multidimensional dynamic systems. Based on the developed structure, new special scoring functions,

новые специальные рейтинговые функции, способы кодирования векторов управления и способы соответствующей дискретизации пространства состояния. Была не только продемонстрирована возможность успешного решения задач оптимального управления поисковыми алгоритмами, но и показаны преимущества в эффективности новых алгоритмов по сравнению с существующими при решении тестовой задачи.

the methods of vector encoding control and state space discretization are proposed. Furthermore, through the test problem solved the capability of such algorithms to solve the optimal control problems, as well as the improved efficiency over the existing algorithms, is demonstrated.

1. *Dijkstra E. W.* A note on two problems in connexion with graphs. – Numerische Mathematik. – 1959. – N 1. – P. 269–271.
2. *Nilsson N.J.* Problem solving methods in artificial intelligence. – McGraw Hill, 1971. – 225 p.
3. *Koenig S.*, *Likhachev M.* Real-Time Adaptive A* // Proc. of the Intern. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS). – 2006. – P. 281–288.
4. *Pinch E.R.* Optimal Control and the Calculus of Variations. – Oxford: University Press, 1993. – 240 p.
5. *Apostolyuk V.* Application of State-Space Search Algorithms to Optimal Control // Proc. of VI Intl. Conf. on Gyro Technology, Navigation, and Motion Control. – K., 2007. – Vol. 2. – P. 104–110.

Рекомендована Радою НАЦ критичних технологій навігаційного приладобудування НТУУ “КПІ”

Надійшла до редакції
4 лютого 2008 року