УДК 519.72

Негадайлов П.А., к. ф.-м. н., асистент
Єщенко А.Ю., студент

Пошук максимальної коаліції в обмеженій кооперативній грі з багатьма гравцями

Київський національний університет імені Тараса Шевченка, 03680, м. Київ, пр-т Глушкова 4д.

P.A. Negadailov, Assistant Professor, Ph. D
A.Y. Yeschenko, student

On the maximum coalition in limited cooperative game with many players

Taras Shevchenko National University of Kyiv, 03680, Kyiv, Glushkova av., 4d
e-mail: pnegadailov@gmail.com, mazepa007@gmail.com

В даній роботі роглядається задача обмеженої кооперативної гри з N гравцями, де можливі коаліції задаються графом зв'язків. Описаний алгоритм пошуку максимальної коаліції.

Ключові слова: кооперативна гра, коаліція, гібридний генетичний алгоритм.

The traditional assumption in cooperative game theory is that every coalition is feasible and can form to attain its payoff (see Mayerson[4]). However, in many real life situations not every group of players has the opportunity to cooperate and to collect their own payoff. In this paper we deal with restrictive cooperative game, where not all coalitions can be formed. There have been previous models developed to study the problem of games with partial cooperation. Games restricted by a communication graph were introduced by Myerson [3] and Owen [4]. Our goal is to find permissible coalition of maximum value. This task is very similar to the maximum clique problem from graph theory which is NP-hard (see, for example, Dinur [2]). That means that it will be interesting to find at least an approximate solution in a reasonable time. For this purpose we developed an hybrid genetic-type algorithm. To substantiate its performance, we conducted a series of experiments on different size graphs.

Keywords: cooperation game, coalition, hybrid genetic algorithm.

Статтю представив д.т.н., проф. Гаращенко Ф.Г.

Кооперативною грою (V,ν) називають функцію $\nu: 2^V \to \Re, \nu(0) := 0$. Гравці є елементами скінченної множини $V = \{v_1, v_2, ..., v_n\}$, а коаліцією є будь-яка підмножина $S \subseteq V$. Розглянемо граф $G = (V, E)$, де V - множина вершин, $E \subseteq V * V$ множина ребер. Будемо казати, що два гравці v_i та v_j можуть належати одній коаліції, якщо ребро $(v_i, v_j) \in E$. Кооперативна гра, в якій можливі не всі коаліції, називається обмеженою кооперативною грою з багатьма гравцями. Подібні задачі розглядалися в роботах [3,5].

На множині гравців задамо невід'ємну функцію $f: V \to R^+$, що характеризує вклад кожного окремого гравця ту чи іншу коаліцію. Задача про максимальну коаліцію полягає у максимізації загальної ваги вершин підграфу:

$$V^* \subseteq V : \sum_{v_i \in V^*} f(v_i) \to max,$$

зберігаючи властивість, що кожні дві вершини з множини V^*, мають містити ребро у множині E (тобто V^*, це обов'язково повний граф):

$$V^* \subseteq V, \forall v_i, v_j \in V^* : (v_i, v_j) \in E$$

Легко бачити. що подібна проблема є аналогом задачі пошуку максимальної кліки (повного підграфу) у зваженому графі, що, як відомо, належить до класу NP-повних, а тому цікавим є питання розробки алгоритму пошуку хоча б наближеного розв'язку задачі за прийнятний час. Для досягнення цієї мети було вирішено застосувати алгоритми генетичного типу.

Проте для задачі пошуку максимальної коаліції, спроба адаптувати класичний

генетичний алгоритм не є продуктивною. Класичні оператори мутації, схрещування і прості випадкові оптимізації не дають хороших результатів через специфіку задачі. Будь-яка зміна хромосом, що представляє повний граф цими операторами, часто веде до кардинальної його зміни та руйнування повноти. Також великою з проблем є розробити ефективний алгоритм для визначення функції пристосовності. Для звичайного генетичного алгоритму, потрібна функція пристосовності, що враховує повноту графу, тобто значення функції має перераховуватися як еквівалент повноти. Проблеми що виникають при таких підходах - граф часто не повний, що веде до складності перевірки на повноту.

Іншою можливістю стало застосування евристичного рандомізованого алгоритму, який зможе гарантувати повноту графу, тим самим елімінуючи проблему з розробкою функції пристосовності. Гарантуючи повний грав на кожній ітерації генетичного алгоритму, значення пристосовності оптимально обирати за сумарною вагою вершин що входять у підграф.

Запропонований нами алгоритм включає в себе жадібну евристичну процедуру. Будуючи максимальну кліку, ми спочатку випадково збільшуємо граф додаючи випадкові вершини, потім ми робимо граф повним, і в кінці спроба збільшення графу, без порушення його повноти.

Перед тим, як описувати розв'язок задачі нагажаємо деякі стандартні означення, пов'язані з генетичними алгоритмами:

Популяція - це множина хромосом X.

Хромосома - це вектор $x=(x_1,x_2,...,x_n)$, $x_i\in\{0,1\}$. $x_i=1$ означає що підграф містить i-ту вершину з основного графа.

Локус - це позиція x_i.

Всі операції алгоритму здійснюва-тимуться на хромосомах. Відомо, що для опису генетичного алгоритму достатнью показати роботу оператора схрещування та мутації. В нашій задачі були використані багатоточкове схрещування та мутація.

Багаточкова мутація

$$if(x_i=1)\to x_i:=0$$
$$if(x_i=0)\to x_i:=1$$

Ця операція виконується для вибірки з n випадкових локусів.

Мотивацією цієї операції є збільшення ймовірності отримати локально правильний розв'язок, тобто можливість потрапити на вершини, які справді знаходяться у максимальній кліці. При розріджених графах він може добавити вершину, що ще не була у розгляді, тоді коли у близьких до повного, може видалити вершину, що може спровокувати неповність підграфу.

Приклад одноточкової мутації: $(1,0,0,0)\to(1,0,1,0)$.

Багатоточкове схрещення

На вході два вектори x і y. Вибираємо вектор випадкових, впорядкованих за зростанням цілих чисел $g=(g_1,g_2,...,g_n)$, $g_i\in(1,|V|)$, $i\in(1,n)$

Створюємо нову хромосому z за таким правилом -локуси з непарних проміжків беремо з x, з парних з y:

$$z=(x_0,x_1,..x_{g_1},y_{g_1+1},y_{g_1+2},..y_{g_2},x_{g_2+1},x_{g_2+2},...,y_{g_{n-1}+(g_n-g_{n-1})-1},y_{g_n})$$

Приклад схрещення: $x=(0,1,0,0,1,1,1,0)$, $y=(1,1,1,1,0,0,0,0)$, обрано вектор $g=(3)$ (одноточкова мутація).

Вектор на виході: $z=(0,1,0,1,0,0,0,0)$

Функція пристосовності

На вході хромосома x, що представляє повний підграф графа V.

$$Fvalue:=\sum_{\forall x_i=1} f(v_i), v_i\in V$$

Евристична процедура

Отримуючи вектор x на вхід ми виконуємо дві процедури. Перша - це приведення до обов'язково повного графу.

- **Ремонт:**

Виберемо випадковий локус у хромосомі ind

(а) **Для кожного** x_i **з** $i=ind$ **до** N: **якщо** $x_i=1$:

* **Або** $x_i:=0$

* **Або з** $j=i+1$ **до** N:

якщо ($x_j=1$ **і** $(v_i,v_j)\notin E$) $x_j:=0$

з $j=1$ **до** $i-1$:

якщо ($x_j=1$ **і** $(v_i,v_j)\notin E$) $x_j:=0$

(b) **З** $i = ind - 1$ **до** 1: **якщо** $x_i = 1$:

* **Або** $x_i = 0$

* **Або з** $j = i - 1$ **до** 1:

якщо ($x_j = 1$ **і** $(v_i, v_j) \notin E$) $x_j = 0$

Легко показати те, що результатом цієї процедури це повний граф. Не залишиться вершин які не зв'язані власним ребром із кожною з хромосоми. Після цієї процедури, завдяки великій залежності від випадковості, може бути видалено багато вершин, що насправді входять у повний підграф. Тому використовується наступна процедура, розширення графу без втрати його повноти.

• **Розширення:**

Виберемо випадковий локус у хромосомі ind

- **Для кожного** x_i **з** $i = ind$ **до** N:

якщо $x_i = 0$ **і** $\forall x_j = 1, j = \overline{1,n} : (v_i, v_j) \in V$:

$x_i := 1.$

- **Для кожного** x_i **з** $i = 1$ **до** $ind - 1$:

якщо $x_i = 0$ **і** $\forall x_j = 1, j = \overline{1,n} : (v_i, v_j) \in V$:

$x_i := 1.$

Ця операція теж дуже залежить від випадкового вибору початкового місця. Після виконання, ми отримуємо обов'язково повний граф, до якого не можна додати жодної вершини, не руйнуючи повноту.

Загальний генетичний алгоритм

Розпишемо детально загальний алгоритм:

Ініціалізація X (локуси хромосом заповнюються значеннями з $\{0,1\}$ з ймовірністю $1/2$)

While(не виконується умова) **DO**

Виконуємо **багатоточкове схрещення**.

Виконуємо **багатоточкову мутацію**.

Виконуємо **евристичну процедуру**.

Виконуємо **оцінювання пристосов-ності** і сортуємо по спаданню її значенн.

Залишаємо кращі $1/3$ значень і додаємо ще $1/6$ значень з проміжку $(4/6,5/6)$ відсортованого вектору.

Оцінюємо умову для виходу з циклу.

END

За допомогою такого підходу, алгоритм не зациклюється на локальних максимумах, і за достатньо малу кількість ітерацій збігається до точного розв'язку. Можна довести що час однієї ітерації $O(n^2)$, що є дуже непоганою альтернативою для точних алгоритмів що потребують $O(2^{n/3})$ часу для незваженої версії. Далі наведені дані з тестування.

Порівняння з алгоритмами перебору

Для оцінки точності розв'язку були проведені порінняння отриманих результатів з точним роз'язком (знайденим за допомогою алгоритму перебору).

Генеруємо спочатку граф на 10 вершин. З коефіцієнтом розрідженості 10 (кожна вершина має максимально 9 ребер, 5 мінімально).

Протестувавши цей граф програмою, отримаємо такі результати:

	t	$W_{\max}$	V_{opt}	V
Перебір	**6 мс**	366	5	10 20
Генетичний	**15 мс**	366	5	10 20

де t - середній час роботи алгоритму, $W_{\max}$- вага максимальної знайденої каоліції. V_{opt} - кількість вершин в максимальній коаліції. V - кількість верщин графа. Остання колонка таблиці відображає кількість запусків алгоритму.

Отже, на дуже малих графах, алгоритм працює швидко як точний так і евристичний. У 20 випадків з 20 знайшов правильну відповідь.

Змінимо граф:

	t	$W_{\max}$	V_{opt}	V
Перебір	**12373 мс**	539	7	20 10
Генетичний	**23 мс**	539	7	20 10

Можна помітити, що час виконання вже відрізняється суттєво. Також змінились результати, у середній кількості вершин, і результат у функції пристосовності (у двох випадках був трішки більший).

Змінимо граф третій раз. Згенеруємо граф з 25 вершинами:

	t	$W_{\max}$	V_{opt}	V
Перебір	**590928 мс**	428	8	25 5

Генетичний **50 мс** 428 8 25 15

Цього разу час иконання відрізняється дуже помітно. Якщо згенерувати більший граф, наприклад з 30ма вершинами, час виконання буде занадто великий.

Тестування на графах великих розмірностей

Наш алгоритм дозволяє знаходити розв'язки для графів великих розмірів за відносно невеликий час. Результати тестування алгоритму на близькому до повного графі що містить 200 верши:

	t,с	$W_{\max}$	V_{opt}
1	**4**	1449	21
2	**5**	1477	22
3	**5**	1492	20
4	**4**	1492	20
5	**8**	1493	20

де перша колонка відповідає номеру запуска. Середній результат за 20 запусків:

t	$W_{\max}$	V_{opt}	V
5.2	1458	21	200

Мінімальне значення функції пристосовності (сумарна вага): 1449, максимальне: 1493.

Маючи NP-повну задачу, точний алгоритм для якої є надзвичайно повільним, незважаючи на особливості задачі, вдалося побудувати ефиктивний евристичний рандо-мізований алгоритм. Експериментальні резльтати показали, що швидкість роботи алгоритму дозволяє застосовувати його для рафів розмірів, отримуючи при ціому розв'язки дуже близькі до точних.

Шляхами до розвитку алгоритму можуть бути, зважена випадковість при видаленні вершин в другому кроці (ремонті). Також, можливо, зміна стандартних операцій мутації, може надати швидшу збіжність до точного розв'язку.

Список використаних джерел

1. E. Algaba. Algorithms for Computing the Myerson Value by Dividends / J.M. Bilbao, J. R. Fernandez, N. Jimenez, J.J. Lopez // Discrete Mathematics Research Progress. – 2007 – P. 1—13.
2. I. Dinur. On the hardness of appro-ximating minimum vertex cover / S. Safra // Annals of Mathematics. – №162. – 2005. – P. 439–485.
3. R.B. Myerson. Graphs and cooperation in games // *Math Oper. Res.* – №2. – 1977. – P. 225–229.
4. R.B. Myerson. Game theory. Analysis of conflict // *Harvard university press. – Cambridge, Massachusetts. London. England*. – 1991. – 568 P.
5. G. Owen. Values of graph-restricted games // *SIAM J Algebraic and Discrete Methods.* – №7. – 1986. – P. 210–220.

References

1. ALGABA E., BILBAO J.M., FER-NANDEZ J.R., LOPEZ J.J. (2007) Algorithms for Computing the Myerson Value by Dividends. Discrete Mathe-matics Research Progress – pp. 1-13.
2. DINUR I. and SAFRA S. (2005) On the hardness of appro-ximating minimum vertex cover. Annals of Mathematics. – 162. – 2005. – pp. 439–485.
3. MYERSON R.B. (1977) Graphs and cooperation in games. *Math Oper. Res.* – 2. – pp. 225–229.
4. MYERSON R.B. (1991) Game theory. Analysis of conflict. *Harvard university press. – Cambridge, Massachusetts. London. England*. – 568 P.
5. OWEN G. (1986) Values of graph-restricted games *SIAM J Algebraic and Discrete Methods.* – 7. – pp. 210–220.

Надійшла до редколегії 23.09.2014