Методика за декларативни CNP-реализации на алгоритми за търсене в Пространство на състоянията

Емилия Големанова

Methodology for Declarative CNP-Implementations of State Space Search Algorithms: *Search is a basic concept of Control Network Programming (CNP), which is a new declarative programming style developed by a team in which the author is involved. The central idea of CNP is that the program is a set of graphs and the "execution" of the program is essentially a process of searching. This and the rich set of built-in tools for search control allow "automatic" (or declarative, non-procedural) implementation of heuristic and stochastic algorithms and to control some basic elements of the computation (number of solutions, depth of recursion, etc.). This makes CNP powerful and efficient tool for search in state space. Presenting the methodology for declarative CNP-implementation of state space search strategies is the main purpose of this paper.*

Key words: *Control Network Programming, CNP, SPIDER, nondeterministic programming, declarative programming, State Space Search Algorithms, Backtracking.*

ВЪВЕДЕНИЕ

Програмирането чрез Управляващи Мрежи (***Control Network Programming, CNP***) е сравнително нова програмна парадигма, която комбинира и разширява възможностите на императивното програмиране, декларативното програмиране, системите, базиращи се на правила и визуално-графичното програмиране. Основно схващане е, че програмата се задава като множество от графи (**Управляваща мрежа**, ***УМ***), а стрелките им са етикетирани с т. нар. **примитиви**, които са елементарни действия (или условия), изпълнявани върху данните. Те се задават като подпрограми (процедури) на някакъв популярен императивен език. Това означава, че потребителят дефинира собствен език от високо ниво за описанието на своя проблем, използвайки функции, които са най-подходящи за собствените му цели. Управляващите структури на този език съответстват на законите за описание на УМ. Разглеждайки процедурите като елементарни единици, езикът е непроцедурен (УМ е всъщност недетерминирано описание на проблема), а „изчислението“ е всъщност търсене на път между начална и крайна точка в УМ. Освен това, CNP разполага с вграден богат инструментариум за управление на това търсене.

Този доклад изследва възможността за използване на „търсенето” в CNP за реализиране на „търсенето” в Пространство на състоянията (ПС) и демонстрира CNP като удобно и мощно средство за реализиране на различни стратегии за търсене. Особено интересни са имплементациите на *Backtracking*-базирани алгоритми, тъй като самото изпълнение на УМ е всъщност **разширен Backtracking**. Този подход на CNP-реализация не изисква разработването (в общоприетия му смисъл) на алгоритъм за търсене – поведението му се постига „автоматично”, благодарение на вградения механизъм и средствата за неговото управление. Докладът е посветен на методиката за приложение на този подход.

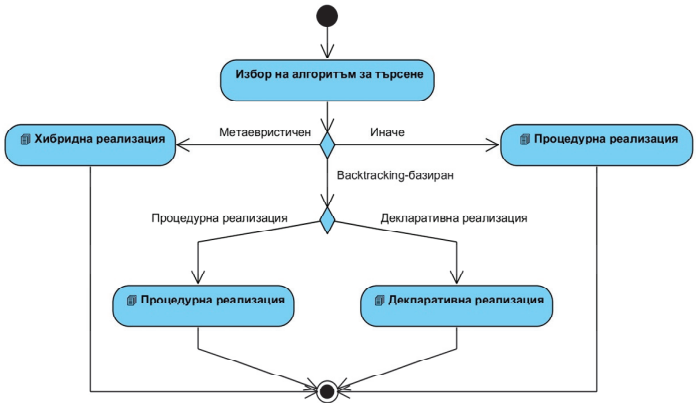
ОСНОВНИ ТЕХНИКИ ЗА РЕАЛИЗАЦИЯ НА АЛГОРИТМИ ЗА ТЪРСЕНЕ В ПС ЧРЕЗ ИЗПОЛЗВАНЕ НА CNP

Алгоритмите за търсене в ПС са базови в областите Изкуствен интелект, Теория на оптимизацията, Дизайн на алгоритми, Логическо програмиране и др.

Този доклад представя възможностите на CNP за програмиране на тези алгоритми в общи схеми (подходи). Разработени са две основни техники [1][2][3][4] и са създадени съответните методики за използването им. При първия подход, класическите алгоритми всъщност се симулират в CNP. Тези реализации са наречени **процедурни реализации**, а CNP се проявява като универсална програмна

парадигма. Много по-различен и по-интересен подход предлага CNP, когато се използва като декларативна програмна парадигма. В този случай УМ просто описва задачата и не се изисква разработването на експлицитна процедура, реализираща процеса на търсене. Вместо това, вграденият механизъм на търсене, заедно с богатата палитра от управляващи го системни средства, извършват избраната стратегия. Тези „автоматични" реализации на алгоритми за търсене са наречени **декларативни** или **непроцедурни** [5]. Не всички стратегии обаче са подходящи за такова елегантно непроцедурно решение. Алгоритмите, които могат директно да бъдат имплементирани трябва да бъдат базирани на стратегията за търсене в дълбочина с връщане при неуспех, т.е. трябва да са *Backtracking*-подобни. Някои по-сложни стратегии, като метаевристичните алгоритми например, изискват комбинация от двете техники – процедурна и декларативна и реализациите им са наречени **хибридни**.

Така, методиката за реализация на различни алгоритми за търсене в ПС започва с избор на вида на реализацията в зависимост от естеството на разработвания алгоритъм (Фиг. 1).



Фиг. 1. *Избор на реализация на алгоритъм за търсене в ПС*

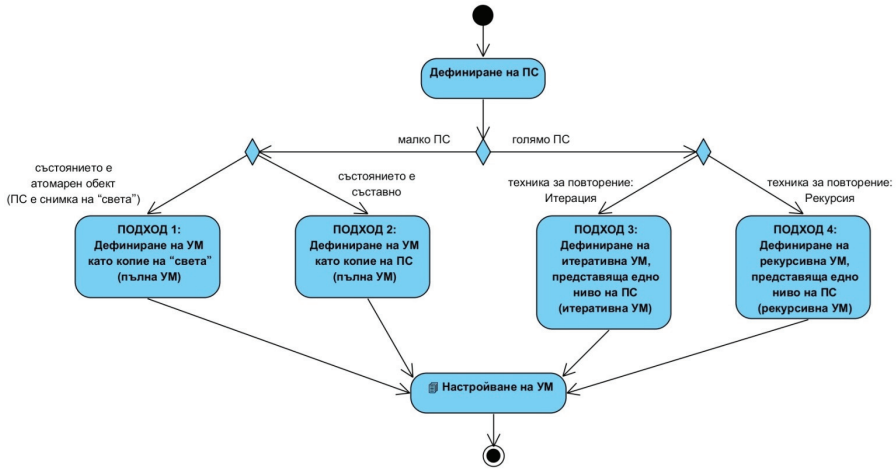
Този доклад е посветен само на декларативната техника за реализация, процедурната е описана в [4].

ДЕКЛАРАТИВНА РЕАЛИЗАЦИЯ НА АЛГОРИТЪМ ЗА ТЪРСЕНЕ В ПС

За създаването на декларативни реализации на алгоритми за търсене е необходимо използването на вградения CNP-механизъм на търсене, който е **разширена версия на стратегията търсене в дълбочина с връщане при неуспех** (**Backtracking**). Тогава е възможно използването на УМ, която има дескриптивен характер (а не процедурен, както при процедурните решения) и задача на вградения интерпретатор е да „изчисли" тази УМ, търсейки път между началния и финален възел. С други думи, ако стратегията, която се моделира е *Backtracking*-базирана, няма нужда от разработване на процедура за търсене – проблемът се описва декларативно (и недетерминирано) чрез дефиниране на неговото ПС, а решаването му се възлага на вградения механизъм на извод в CNP. Методиката за създаване на непроцедурни реализации на алгоритми за търсене в ПС е представена на Фиг.2.

Разработени са **четири основни подхода** в зависимост от вида и размера на ПС и избраната техника за реализиране на повторение – итерация или рекурсия. Всеки подход може да се разглежда като обща схема (шаблон, *framework*) за

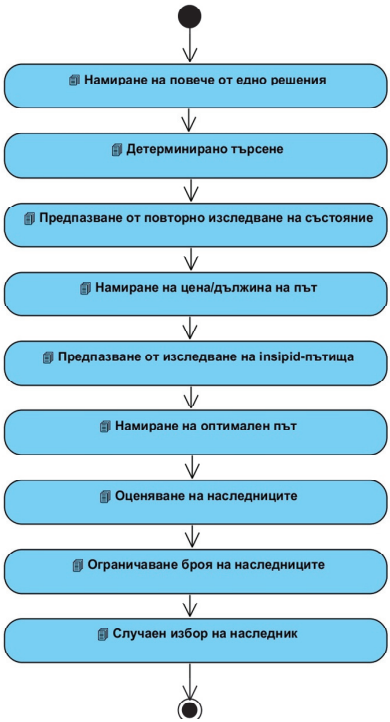
решаване на различни задачи, но с еднакви характеристики на тяхното ПС. От друга страна, всеки алгоритъм, еволюиращ от стратегията *Backtracking*, би могъл да бъде имплементиран декларативно, използвайки и четирите подхода. Това прави методиката алгоритмично и проблемно-независима.



Фиг. 2. *Методика за разработване на декларативна реализация на алгоритми за търсене в ПС*

Във всички предложени декларативни CNP-решения се използва богата палитра от средства за статично и динамично управление на търсенето в УМ – т.нар. **системни опции** и **управляващи състояния**. Прилагането им се нарича настройване на УМ съобразно изискванията и спецификата на разработвания алгоритъм. Методиката за **настройването на УМ** е представена на Фиг. 3, като отделните нейни елементи са описани по-подробно в [4].

Използвайки вградените средства за контрол на изчислението, програмистът може лесно да моделира различни режими на избор на излизаща стрелка от дадено състояние на УМ, която да се изпробва първо. Например изборът на стрелка може да е случаен или да се изследва най-напред най-перспективната, или тази, която съвпада с дадена стойност, или пък тази, която попада в зададен оценъчен интервал и др. Принципно, CNP-програмистът може да „прави каквото си иска" със стрелките, излизащи от дадено състояние – да ги преподрежда и/или ограничава техния брой. Управлението се „придвижва" от това състояние към следващото по дадена стрелка, но може и да се „върне" и да изследва друга излизаща от него стрелка. Това прилича на подхода, използван в Локалното търсене, с тази разлика, че в CNP се пази и изминатия път, с цел връщане при неуспех. Затова CNP позволява лесно и естествено (интуитивно) да се реализират непроцедурни решения на Локално търсене и други алгоритми в основата, на които стои идеята на стратегията *Backtracking*. Освен преподредбата и ограничаването на броя на излизащите стрелки, програмистът може да модифицира и някои други параметри на търсенето, като например да забрани възвратa, да ограничи дължината/разходите на пътя на решение или дълбочината на рекурсията/итерацията и др.



Фиг. 3. *Методика за настройване на УМ, съобразно изискванията на избрания алгоритъм*

Така дадена УМ може да бъде настроена съобразно характеристиките и изискванията на разработвания алгоритъм. По този начин едно CNP-решение, реализиращо даден алгоритъм лесно може да бъде модифицирано до друг алгоритъм или до същия, но с други параметри, пренастройвайки управляващата му мрежа. Това определя предлагания метод за разработване на декларативни имплементации чрез CNP като адаптивен и гъвкав.

Четирите основни подхода за разработване на декларативни CNP-решения чрез използването на вградените средства за управление на изчислението са описани в [4]. Реализирани са както евристични, така и усъвършенствани неинформирани алгоритми за търсене в ПС, произлизащи от *Backtracking*-стратегията. Предложени са непроцедурни имплементации на:

- ***Backtracking***;
- ***Cost-Limited Search*** и ***Depth-Limited Search***;
- **Търсене на оптимално решение** с отрязване на *insipid*-пътища (версия на ***Branch-and-Bound***);
- ***Hill-Climbing***;
- ***Irrevocable Hill-Climbing*** и ***Nearest Neighbour Search***;
- ***Hill-Climbing*** **с ограничен възврат**;
- ***Stochastic Hill-Climbing***;
- ***First-Choice Hill-Climbing***,

като тук ще бъде разгледана само тази на *Backtracking*-алгоритъма.

Разработването им е проследено съобразно методиката от Фиг. 2 и Фиг. 3 върху известни игрови задачи от Изкуствения интелект, често използвани като тестови за изследване на различни алгоритми за търсене в ПС.

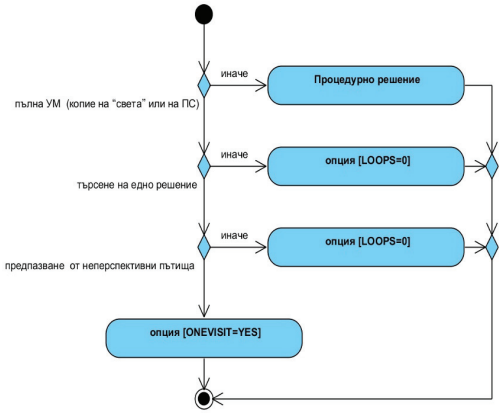
ДЕКЛАРАТИВНА РЕАЛИЗАЦИЯ НА АЛГОРИТЪМ *BACKTRACKING*

Ако се моделира алгоритъмът *Backtracking*, няма нужда от процедурното му имплементиране – проблемът просто се описва, а решаването му се предоставя на вградената в SPIDER машина за извод. Нещо повече, тъй като този алгоритъм не е евристичен, не се използва и инструментариумът на езика за динамичен контрол на търсенето. На CNP-програмиста, евентуално, може да му се наложи използването само на средствата за статично управление с цел предпазване от попадане в цикъл или специфициране на максимален брой решения, максимална дължина/разходи на пътя на решение и др. Тази идея е представена в CNP-приложенията, описани в [6], решаващи задачи с вътрешно-присъщо графоподобно представяне (задачата за идентификация на животни, задачата за разпознаване на аритметичен израз и др.), а тук ще бъде развита за решаване на абстрактната задача за търсене на път в ПС п метода *Backtracking*. Нейната универсалност е демонстрирана в [4] върху четири игрови задачи, но тук ще бъде представена само за задачата за маймуната и банана (*Monkey and Banana Problem*) [7]:

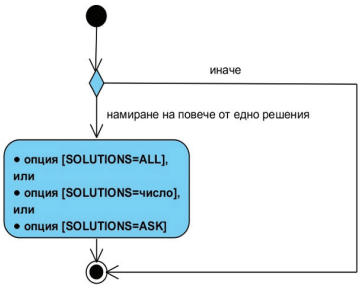
В стая, до вратата, има маймуна. В средата на стаята, на тавана е закрепен банан. Маймуната е гладна и иска да вземе банана, но не може да го стигне. До прозореца на стаята има сандък и маймуната може да го използва. Маймуната може да извършва следните действия: да се движи в стаята; да се качва на сандъка; да мести сандъка, ако е до него; да вземе банана, ако се намира върху сандъка, точно под банана.

Според методиката от Фиг. 2 началната стъпка в разработването на декларативни CNP-решения е дефинирането на ПС на конкретната задача. Ако то може да бъде зададено явно, т.е. ПС е с приемлива големина, от гледна точка съответстващата му УМ, то цялото ПС се конвертира в УМ почти тривиално – на състояние от ПС се съпоставя състояние от УМ, а на оператор от ПС – последователност от примитиви на УМ, кодиращи логиката на този оператор. Такава УМ ще бъде наричана **пълна**. Подходите, създаващи пълна УМ са два (Подход 1 и Подход 2) в зависимост от вида на състоянието на ПС. В случай, че ПС е голямо и се налага то да бъде неявно зададено (чрез функцията за генериране на наследници), се използват други два подхода за създаване на декларативно CNP-решение, които се различават по използваната техника за реализиране на повторение. Също както при първите два подхода, УМ декларативно представя задачата, но не „копира" цялото ПС, а само част от него (едно ниво) и цялостното решение се постига използвайки итерация или рекурсия. За такъв вид УМ е използвано понятието **итеративна УМ**, съответно **рекурсивна УМ**. Подход 3 от представяната методика е в основата на всички CNP-решения с итеративна УМ, а Подход 4 – на тези с рекурсивна УМ. Тези два подхода са универсални и могат да се приложат върху задачи с произволен размер на ПС, но са особено подходящи за тези с големи ПС.

След дефинирането на УМ следва нейното настройване според методиката от Фиг. 3. Тази методика описва последователност от действия, съответстващи на възможните изисквания на разработвания алгоритъм. Например типично изискване към решението е да намира всички ациклични пътища в ПС. Това налага използването на методиките за „**Предпазване от повторно изследване на състояние от ПС**" и „**Намиране на повече от едно решения**", представени съответно на Фиг. 4 и Фиг. 5.



Фиг. 4. *Методика за CNP-реализация на механизъм за предпазване от повторно изследване на състояние от ПС*



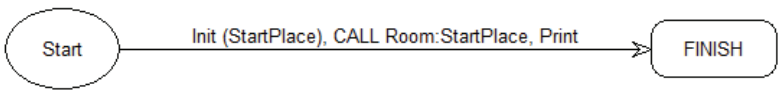
Фиг. 5. *Методика за CNP-реализация на механизъм за намиране на повече от едно решения*

Като пример за приложение на описаната методика на Фиг.6 е представено само едно решение на задачата за маймуната и банана, използвайки **Подход 1: Пълна УМ, представяща простото ПС на задачата.** Този подход може да бъде приложен върху задачи, специфицирани чрез т.нар. **просто** ПС [8]. При него състоянието няма вътрешна структура, а е атомарен обект (**token**). Характерно за този вид ПС е, че то представя физическата среда на задачата, т.е. е „**снимка на света**" [9]. Подобно ПС имат т.нар. „задачи за маршрутизиране" (*routing problems*).

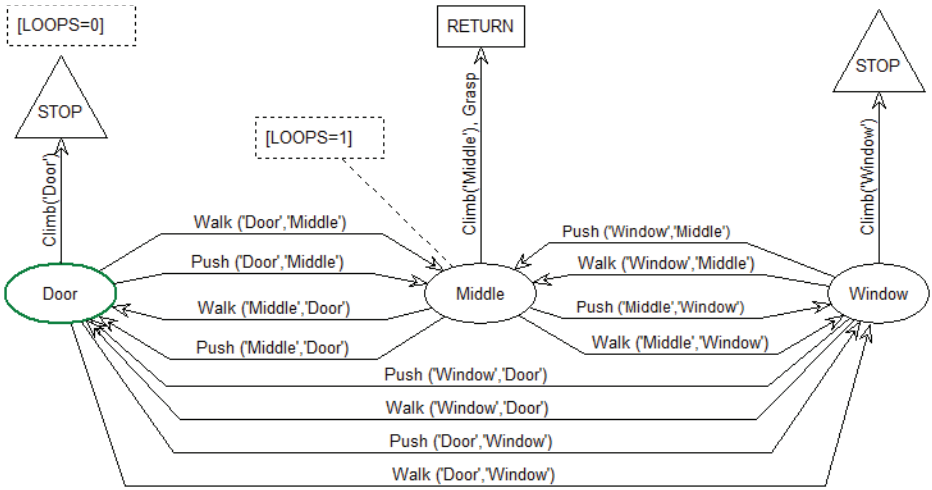
Простото ПС на задачата за маймуната и банана е „картата" на стаята. Състоянието е позиция в стаята (*Door*, *Middle* или *Window*), а операторът – възможно действие на маймуната от дадено състояние (*Walk*, *Push*, *Climb* или *Grasp*). Подмрежата ***Room*** от Фиг. 4 "копира" това ПС:

[SOLUTIONS=ALL]
MAIN MonkeyAndBanana;

```
var StartPlace : string
```



SUB Room;



Фиг. 6. *Декларативно CNP-решение на задачата за маймуната и банана – УМ съответствия на картата на стаята*

Това решение предполага търсенето на всички ациклични пътища от позиции на маймуната. Затова са използвани опциите [SOLUTIONS=ALL] и [LOOPS=0], като само за състояние *Middle*, опцията е [LOOPS=1]. Това позволява началната позиция на маймуната да бъде *Middle*.

ЗАКЛЮЧЕНИЕ

В заключение, представеният декларативен подход за реализация на алгоритми за търсене в ПС се характеризира с това, че **УМ просто описва задачата** и не се изисква разработването на експлицитна процедура, реализираща процеса на търсене. Вместо това, вграденият механизъм на търсене, заедно с богатата палитра от управляващи го системни средства, извършват избраната стратегия. В резултат, тези „автоматични" реализации на алгоритми за търсене са **интуитивни** и **естествени** и съответстват на човешкия начин на мислене и специфициране на сложни проблеми. Вградените средства за статично и динамично управление на механизма на търсене в CNP се оказват много **удобен** и **мощен инструмент** за реализация на богат набор от слепи и евристични стратегии от ПС, както и на метаевристични алгоритми (Симулирано закаляване) от Теория на оптимизацията. Освен това едно CNP-решение, реализиращо даден алгоритъм лесно може да бъде модифицирано до друг алгоритъм или до същия, но с други параметри, пренастройвайки управляващата му мрежа чрез използване на други системни средства. Това определя предлагания метод за разработване на декларативни имплементации чрез CNP като **адаптивен** и **гъвкав**.

ЛИТЕРАТУРА

[1] K. Kratchanov, E. Golemanova, T. Golemanov, and Y. Gökçen, "Implementing Search Strategies in Winspider II: Declarative, Procedural, and Hybrid Approaches," in Knowledge-Based Automated Software Engineering, I. Stanev and K. Grigorova, Eds. Cambridge Scholars Publishing, 2012, pp. 115–135.

[2] K. Kratchanov, E. Golemanova, T. Golemanov, T. Ercan, and B. Ekici, "Procedural and Nonprocedural Implementation of Search Strategies in Control Network Programming," in Intern. Symposium on Innovations in Intelligence Systems and Applications (INISTA 2010), 2010, pp. 386–390.

[3] K. Kratchanov, E. Golemanova, T. Golemanov, and Y. Gokcen, “Declarative and Procedural Search Strategy Implementations in WinSpider,” in Fundamental Sciences and Applications, Plovdiv, Bulgaria, J. of Technical Univ. at Plovdiv, 2011, pp. v.16, book1, 217–22.

[4] Е. Големанова, “ИЗСЛЕДВАНЕ НА ПАРАДИГМАТА „ТЪРСЕНЕ” И СЪЗДАВАНЕ НА МЕТОДИКА ЗА РЕАЛИЗИРАНЕ НА АЛГОРИТМИ ЗА ТЪРСЕНЕ ПРИ ПРОГРАМИРАНЕТО ЧРЕЗ УПРАВЛЯВАЩИ МРЕЖИ,” РУ “А. Кънчев,” 2014.

[5] K. Kratchanov, E. Golemanova, T. Golemanov, and T. Ercan, “Non-procedural Implementation of Local Heuristic Search in Control Network Programming,” in 14th Int. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2010), 2010, pp. 263–272.

[6] K. Kratchanov, E. Golemanova, and T. Golemanov, “Control Network Programming Illustrated: Solving Problems with Inherent Graph-Like Representation,” in Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008), 2008, pp. 453–459.

[7] I. Bratko, Prolog Programming for Artificial Intelligence, 4th ed. Pearson Education, 2011.

[8] C. Thornton and B. Du Boulay, Artificial Intelligence: Strategies, Applications and Models Through SEARCH. AMACOM, 1998.

[9] A. Barr and E. Feigenbaum, Eds., The Handbook of Artificial Intelligence. HeurisTech Press, William Kaufmann, Inc., 1981.

За контакти:

гл. ас. д-р Емилия Големанова, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 082-888 681, e-mail: EGolemanova@ecs.uni-ruse.bg

Докладът е рецензиран.