25. Търсене в пространството от състояния. Генетични алгоритми.

1. **Основни понятия. Пространство на състоянията**

   Деф: Състояние
   Състояние е представяне(формулиране) на задачата в процеса на нейното решаване

   Има няколко вида състояния: начални, междинни и крайни (целеви) състояния.

   Деф: Оператор
   Начин (правило, алгоритъм, функция,..) по който едно състояние се получава от друго.

   Деф: Пространство на състоянията (ПС)
   Съвкупността от всички възможни състояния, които могат да се получат от дадено начално състояние.

   Деф: Решение
   Решението е последователност от действия, които водят от начално до финално състояние.

   Пространството на състоянията може да се представи чрез (ориентиран) граф или дърво с възли състоянията и ориентирани дъги - операторите.

   Основните действия, свързани с пространството на състоянията са:
   - Генериране на състояния - на следващи наследници или на всички наследници
   - Оценяване на състояния - true/false или числова оценка в определен интервал
   - Дефиниране на цялото ПС

   Деф: Стратегия
   Стратегията е избирането на реда на разширяване на възлите

   Стратегиите се оценяват на база на следните показатели:
   - Пълнота - винаги ли се намира решение, ако съществува такова
   - Времева сложност - брой на генерираните/разширени възли
   - Пространствена сложност - максимален брой възли в паметта
   - Оптималност - винаги ли се намира решение с най-малки разходи

2. **Формулировка на основните типове задачи за търсене в пространството на състоянията**

   Основни типове задачи, свързани с ПС: генериране на ПС, решаване на задачи върху генерирано ПС, комбинирано генериране и търсене в ПС.
   Стратегиите за тези задачи са сходни и затова обикновено се говори само за търсене (а се подразбира и/или генериране)

   ◇ *Търсене на път до определена цел*
   Пътят може да се търси под формата на списък от състояния или дъги (действия) в ПС.
   Вариация е търсенето на минимален път до цел.

   Видове спрямо наличната информация
   - *Неинформирано (сляпо) търсене* - използва се само информацията, налична при дефинирането на проблема. Пример: търсене в дълбочина, в широчина, лимитирано търсене в дълбочина, итеративно задълбочаващо се търсене
   - *Информирано търсене* - стратегиите имат информация за целевото състояние, която им помага за по-ефективно търсене. Тази информация се получава от функция (евристика), която естимира колко близо е състояние до целевото състояние. Примери: A*, Алчно Best-first search, beam search, hill climbing.

   Видове спрямо разглежданото пространство:
   - Глобално търсещи - взимат предвид цялото ПС
   - Локално търсещи - гледат само локална област, ако решението не е в тази област няма да могат да го намерят

Параметри на търсенето:
- Дълбочина на "най-плитката" цел - d
- Максимална "дълбочина" на пространството/графа на състоянията - m
- Коефициент на разклонение на ПС - b

◇ *Формиране на стратегия при игри за двама играчи*
Класическа постановка на задачата:

Разглеждат се т.нар. интелектуални игри с пълна информация, които се играят от двама играчи и върху хода на които не оказват влияние случайни фактори. Двамата играчи играят последователно и всеки има пълна информация за хода на играта. Най-често се решава задачата за намиране на най-добър първи ход на играча, който трябва да направи текущия ход.

Основни типове игри:
- С перфектна/неперфектна информация
- Детерминистични/включващи елементи на шанс игри
- С пълна/непълна информаци

Примерни задачи от този тип: шахмат, морски шах и др.
За решаването им може да се изпомилзва минимакс процедура, минимаксна процедура с алфа-бета отсичане и др.

◇ *Намиране на цел при спазване на ограничителни условия*
Дадени са:
- Множество променливи $v_1, v_2, \dots, v_n$
- Множество от ограничения

Целево състояние (състояния): множество от свързвания със стойности на променливите, които удовлетворяват всички ограничения.

Примерни задачи от разглеждания тип: Задача за осемте царици, оцветяване на географска карта, пъзели, судоку, разписания и др.

За решаването им може да се използва: генериране и тестване, backtracking, разпространяване на ограниченията, локално търсещи алгоритми (min conflicts)

3. Методи за информирано (евристично) търсене на път до определена цел
Приложими са при наличие на специфична информация за предметната област, позволяваща да се конструира оценяваща функция (евристика), която връща като резултат приблизителната стойност на определен ресурс, необходим за достигане от оценяваното състояние до целта.

1. Best-first search
Това е търсене на най-добър път.

Идея: използване на оценяваща функция за всеки възел, за да се оцени неговата "желаност". Разширяване на най-желания неразширен възел.

Имплементация - поддържа се опашка със сортирани възли в намаляващ ред спрямо оценката им за "желаност". (сортира се списъка в съответствие с евристичната функция)

Оценка на метода:
- Ефективен, но не е пълен или оптимален
- Времевата сложност е $O(b^m)$, но подходяща евристика може да доведе до съществено подобрение
- Пространствената сложност е $O(b^m)$, защото се съхраняват всички достигнати състояния

2. **Beam search** (търсене с ограничена широчина / търсене в лъч)
   Алчен best-first search с опашка с ограничен размер n, т.е. ограничава списъка до първите n най-добри възела в него (спрямо евристиката).

   Оценка на метода:
   - Не е нито пълен, нито оптимален (локално търсещ)
   - O(bnm) - Времевата сложност зависи от разгледаното изрязано пространство и евристика. При използване на подходяща евристика може да доведе до съществено подобрение
   - O(bn) - линейна пространствена сложност, зависи от параметъра n

3. **Hill climbing** (метод на най-бързото изкачване)
   Списъкът се ограничава до най-добрият му елемент (в съответствие с евристиката) и то само, ако той е по-добър от своя родител. Търсенето е еднопосочно без възможност за възврат.

   Оценка на метода:
   - Не е нито пълен, нито оптимален (локално търсещ)
   - O(bm) - линейна времева сложност - зависи от дълбочината на пространството и евристиката
   - O(b) - константна пространствена сложност

4. **A*** - търсене с минимизиране на общата цена на пътя
   Идея: Да избягваме да разширяваме пътища, които вече са скъпи. A^* търсенето е оптимално

   Използва се оценяваща функция $f(n) = g(n) + h(n)$
   - $g(n)$ - цена до тук за достигане на n
   - $h(n)$ - приблизителна цена за достигане на цел от n
   - $f(n)$ - очаквана обща цена за път, минаващ през n и достигащ до цел

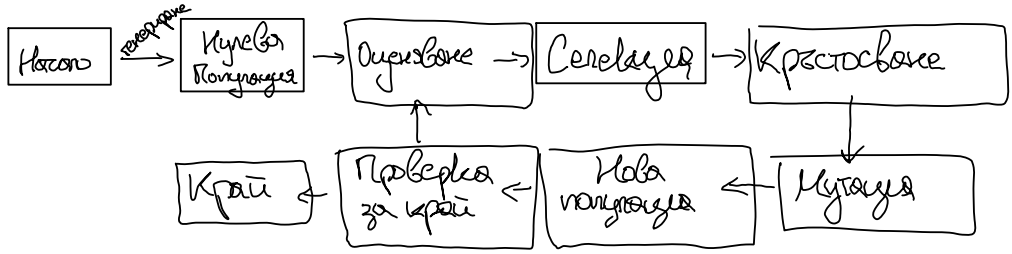
   Използва се евристика $h(n)$ със следните характеристики:
   - $h(n) \leq h^*(n)$, където $h^*(n)$ е реалната цена от n до цел
   - $h(n) \geq 0$, за да може $h(G) = 0$ за всяка цел G
   - т.е. евристиката никога не надценява действителното разстояние

   Оценка на метода:
   - Пълен, освен ако няма безкрайно много възли с $f \leq f(G)$
   - Оптимален
   - Времева сложност $O(b^d)$ - експоненциална
   - Пространствена сложност $O(b^d)$ - държи всички възли в паметт

4. Генетични алгоритми

Вариант на стохастично beam search, при което новите състояния се генерират чрез комбиниране на двойки родителски състояния, вместо чрез модифициране на текущото състояние
Тип локално търсене, което имитира еволюцията като взема популация от стрингове, които кодират възможни решения и ги комбинира на база оценяваща функция (fitness function), за да създава индивиди, които са по-подходящи.

Основни принципи:
- Състоянията се представят като низове над дадена крайна азбука (хромозоми)
- Оценяваща функция (fitness function) оценява пригодността (близостта до целта) на съответното състояние. Има по-големи стойности за по-добрите състояния
- Алгоритъмът започва с множество (популация) от k случайно генерирани състояния (поколение 0)
- Принципи на получаване индивиди за следващото поколение: селекция, кръстосване, мутация



◇ Селекция - етап, през който се избират представители от популацията, които да бъдат използвани за размножаване (чрез кръстосване)

◇ Кръстосване - генетичен оператор, използван за промяна на индивидите от едно поколение към следващото. Аналогичен е на размножаването в биологичен смисъл. Представлява процесът на взимане на повече от едно родителско състояние и създаване на състояние-наследник от тях.

Типове кръстосване: Нека дължината на хромозомите е L

- One-point кръстосване - избира се случайна позиция между 1 и L-1. Хромозомите се отрязват на зададената позиция. Разменят се вторите им половини.

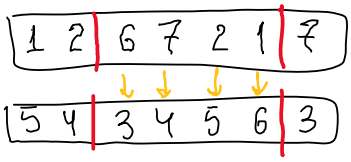
- Order кръстосване
  - Случайно се избират се две срязващи точки
  - Низа между двете точки в първия родител е копиран в наследника
  - Останалите позиции се запълзват, взимайки предвид реда на гените (символите) във втория родител, започвайки след изрязания интервал

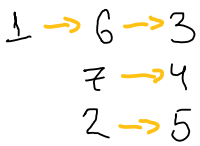
- Циклично кръстосване - идентифицира даден брой "цикли" между двата родителски хромозома. За да се получи наследник 1, цикъл 1 се копира от родител 1, цикъл 2 от родител 2, цикъл 3 от родител 1 и т.н.
  Пример:
  P1: 8 4 7 3 6 2 5 1 9 0
  P2: 0 1 2 3 4 5 6 7 8 9

  Цикли са: 8 → 0 → 0 → 9 → 9 → 8; (Стойности 8, 9, 0)
  4 → 1 → 1 → 7 → 7 → 2 → 2 → 5 → 5 → 6 (Стойности 4, 1, 7, 2, 5, 6)

  Резултатни хромозоми:
  Д1: 8 1 2 3 4 5 6 7 9 0
  Д2: 0 4 7 3 6 2 5 1 8 9

- Patially mapped crossover -
  - случайно се избира интервал за изрязване (две позиции)
  - подниза от интервала в първия родител замества съответстващия подниз във втория родител
  - Определяме релация на съпоставяне спрямо поднизовете в двата интервала. Пример:





  - Използваме съпоставянето, за да направим наследниците коректни (спрямо задачата която решаваме, може да имаме различни инварианти относно хромозомите - като това всички символи да са различни)

- Uniform кръстосване
  - За всяка позиция взима ген (символ) от един от родителите на случаен принцип

◇ Мутация
Извършване на случайни промени в случайно избрана малка част от новата популация с цел да се осигури възможност за достигане до всяка точка от ПС и да се избегне попадане в локален екстремум.

Примери за различни типове:

- Произволно променяне на знак от низа
- Размяна на два знака от низа
- Вмъкване на знак от низа на друга позиция (шифтва останалите)
- Обръщане на редът на символите във вътрешен интервал