Търсене в пространството от състояния. Генетични алгоритми.

Тема 22 *(до 07.2022)* и Тема 25 *(от 09.2022)*

Основни точки по темата:

Пространство на състоянията. Основни понятия. Формулировка на основните типове задачи за търсене в пространството на състоянията: търсене на път до определена цел, формиране на стратегия при игри за двама играчи, намиране на цел при спазване на ограничителни условия. Методи за информирано (евристично) търсене на път до определена цел: best-first search, beam search, hill climbing, A*. Генетични алгоритми - основен алгоритъм , типове кръстосване и мутация.

**** Литература:*** *Stuart Russell & Peter Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall, 2003.*

1. Пространство на състоянията. Основни понятия.

Обща постановка. Решаването на много задачи, традиционно смятани за интелектуални, може да бъде сведено до последователно преминаване от едно описание (формулировка) на задачата към друго, еквивалентно на първото или по-просто от него, докато се стигне до това, което се смята за решение на задачата.

Примери: задачи от областта на т. нар. интелектуални игри, аналитични преобразувания на алгебрични и тригонометрични изрази, решаване на уравнения и т.н.

Основни дефиниции

- **Състояние:** едно описание (формулировка) на задачата в процеса на нейното решаване.
- **Видове състояния:** начално, междинни, крайни (целеви).
- **Оператор:** начин (правило, алгоритъм), по който от дадено състояние се получава друго.
- **Пространство на състоянията (ПС):** съвкупността от всички възможни състояния, които могат да се получат от дадено начално състояние.

6	5	3
2	4	8
7	0	1

Initial State

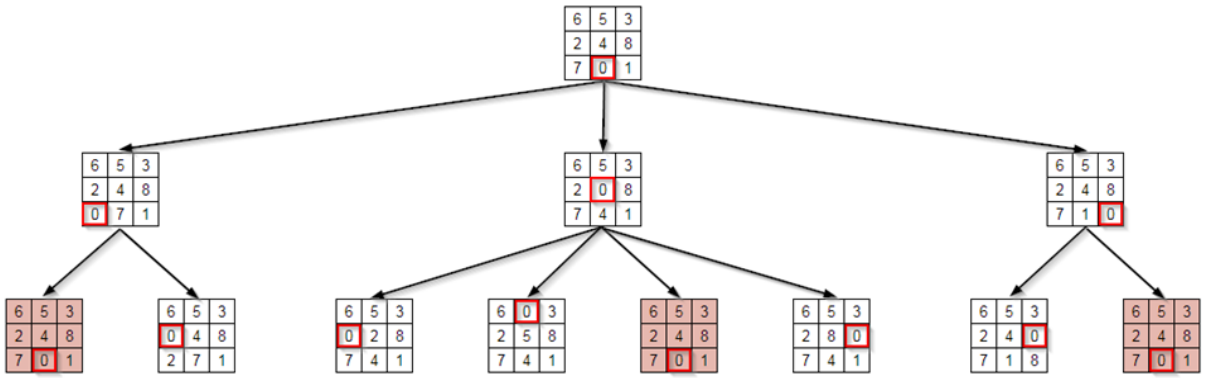
⇒

1	2	3
4	5	6
7	8	0

Goal State

Представяне на ПС: чрез ориентиран граф (граф на състоянията, ГС) с възли – състоянията и дъги – операторите. Когато ПС може да се представи във вид на дърво, се говори за т. нар. дърво на състоянията (ДС).
Действия, свързани с ПС

- Генериране на състояния
  - генериране на следващ наследник
  - генериране на всички наследници
- Оценяване на състояние
  - двоични оценки (true/false)
  - числови оценки в определен интервал (оценката на целта съвпада с единия край на интервала)



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

Основни типове задачи, свързани с ПС

- Генериране на ПС
- Решаване на задачи (търсене) върху генерирано ПС (офлайн)
- Комбинирано генериране и търсене в ПС (онлайн)

Стратегиите за решаване на тези задачи са сходни и затова обикновено се говори само за търсене (а се подразбира и/или генерирането).

2.1. Търсене на път до определена цел.

Търси се път от дадено начално състояние до определено целево състояние (целевото състояние е описано явно или може да бъде разпознато). Пътят може да се търси под формата на списък от възли или списък от дъги в ГС.
Варианти: търсене на минимален (най-къс или най-икономичен) път до цел и др.

- Според наличната информация
  - **Неинформирано (сляпо) търсене:** Depth First Search, Breadth First Search, Depth Limited Search, Iterative Deepening Search, Uniform Cost Search
  - **Информирано (евристично) търсене:** Best First Search, Beam Search, Hill Climbing, A*
- Според разглежданото пространство
  - **Глобално търсещи:** DFS, BFS, IDS, UCS, Best First Search, A*
  - **Локално търсещи:** ~~DLS,~~ Beam Search, Hill Climbing, Simulated Annealing, Genetic Algorithm

  ** Локалното търсене не се прилага за намиране на път от точка А до точка Б, а за оптимизационни задачи, в които целевото състояние се знае как изглежда, но не се знае кое е най-доброто. Използва се евристика, която да насочва към подобряване на крайното състояние. Това прави локално търсещите алгоритми информирани. Затова и DLS е зачеркнат.*

Примерни задачи от реалния свят:

- Намиране на път (при routing в компютърни мрежи, при планиране на пътувания с автомобилен/самолетен транспорт и др.)
- Задача за търговския пътник, навигация на роботи, задачи за планиране и конструиране (асемблиране) на сложни обекти и др.

Характеристики на алгоритмите за търсене:

- **Пълнота:** Един алгоритъм е пълен тогава, когато ако съществува решение, то той гарантира, че ще го намери.
- **Оптималност:** Един алгоритъм е оптимален тогава, когато ако има два или повече пътя, които водят до решение, то той гарантира, че ще намери както най-близкото решение (оптималното), така и най-късия път до това решение.
- **Сложност**
  - **по време** ≈ брой изследвани (обходени) възли
  - **по памет** ≈ максимален размер на фронта (максимален брой възли в паметта)

***** Необходимост от избягване на повторения и зацикляне.*

Параметри на търсенето:

- Дълбочина на “най-плитката” цел – ***d***
- Максимална “дълбочина” на пространството/графа на състоянията – ***m***
- Коефициент на разклонение (разклоненост) на ГС – ***b***

2.2. 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 от тях 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 елемент на шанс* игри
- С *пълна/непълна* информация

Представяне на задачата като дървовидна структура (ДС):
ДС при тези задачи е дърво на възможните позиции в резултат на възможните ходове на двамата играчи. Общият брой възли в ДС е от порядъка на b^d (***b*** - коефициент на разклонение на игровото дърво, ***d*** – височина (дълбочина) на игровото дърво).
Пример. В шахмата е установено, че ***b*** има средна стойност около 35, а всеки играч играе около 50 хода, т.е. ***d*** има средна стойност около 100.

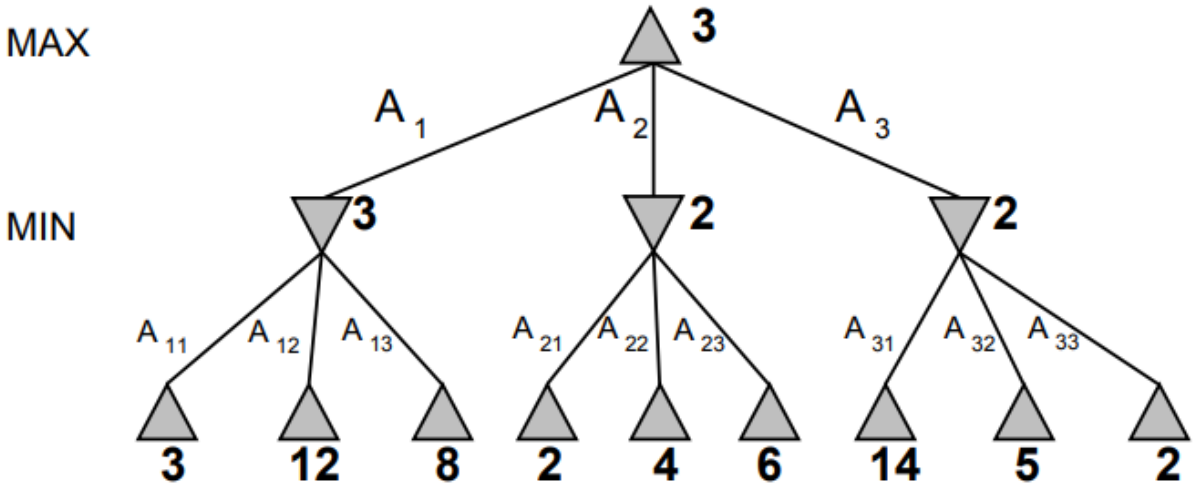
Примерни задачи от разглеждания тип:

- шахмат, шашки, “кръстчета и нулички” и др.

Алгоритми за решаване на задачи от този тип:

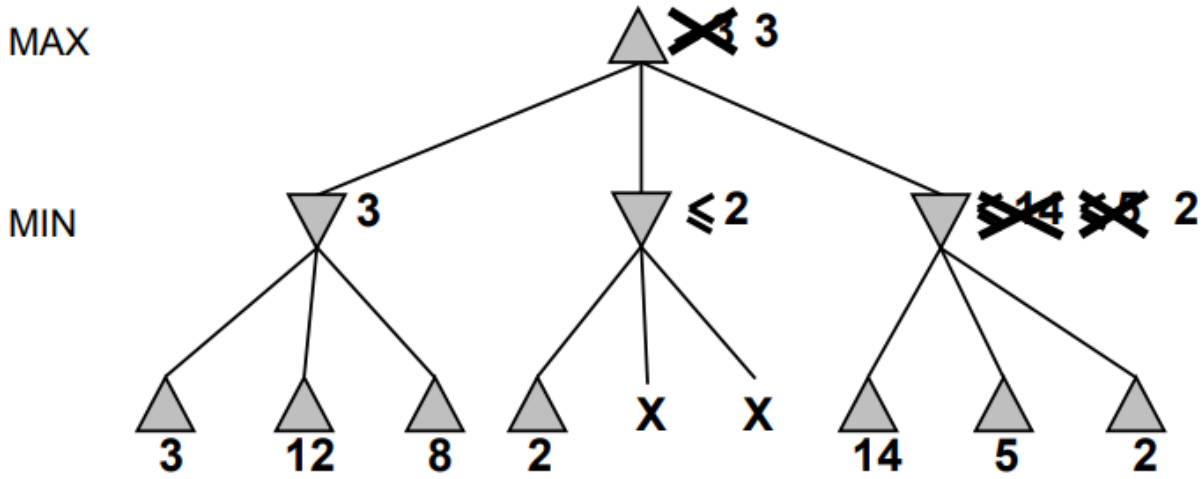
- **Минимаксна процедура:** Метод за намиране на най-добрия ход на първия играч при предположение, че другият играч също играе оптимално. Изисква построяване на цялото ДС и намиране на оценките на листата (наричат се статични оценки).
  - Предполага се, че двамата играчи имат противоположни интереси, изразени в това, че единият търси стратегия, която води до получаване на позиция с максимална оценка (максимизиращ играч, MAX), а другият търси стратегия, която води до получаване на позиция с минимална оценка (минимизиращ играч, MIN).
  - Минимаксната процедура е метод за получаване на оценките (наричат се придобити или породени оценки) на възлите от по-горните нива на ДС, които позволяват на първия играч да избере най-добрия си ход.
  - Оценките се разпространяват отдолу нагоре, като всеки от възлите, съответни на ход на максимизиращия играч, получава оценка, равна на максималната от оценките на преките му наследници, а всеки от възлите, съответни на ход на

минимизиращия играч, получава оценка, равна на минималната от оценките на преките му наследници.
- Оценка на минимаксната процедура изисква построяване на цялото ДС, което може да се окаже твърде голямо в общия случай на реална игра. Точен, но крайно неефективен и често нереализуем на практика метод.



- **Минимаксна процедура с алфа-бета отсичане (Алфа-бета процедура):** Преодоляват се проблемите при минимаксната процедура, породени от обстоятелството, че там процесът на генерирането на ДС е напълно отделен от процеса на оценяване на позициите, което води до силна неефективност.
  - Тук листата се оценяват веднага след генерирането им и при първа възможност се пресмятат и съответните придобити (породени) оценки на възлите от по-горните нива или поне се намират подходящи горни граници на оценките на възлите от минимизиращите нива (съответни на ходове на минимизиращия играч) и/или долни граници на оценките на възлите от максимизиращите нива (съответни на ходове на максимизиращия играч). Така броят на операциите за намиране на същия резултат, както при минимаксната процедура, може да се намали значително.
  - Алфа-бета процедурата изисква генериране на ДС в дълбочина, при което се преценява безполезността от генерирането и оценяването на някои клонове, не оказващи влияние върху резултата.
  - *Дефиниции:* Алфа-стойност на даден максимизиращ възел се нарича текущо установената долна граница на породената му оценка. Бета-стойност на даден минимизиращ възел се нарича текущо установената горна граница на породената му оценка. Първоначално определените алфа- и бета- стойности на възлите могат да се уточняват в процеса на работа, като

алфа-стойностите могат само да растат, а бета-стойностите – само да намаляват.
- *Описание на метода (Правила за прекратяване на генерирането и търсенето):* При алфа-отсичането не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от минимизиращ възел, бета-стойността на който е по-малка или равна на алфа-стойността на съответния максимизиращ родител. При бета-отсичането не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от максимизиращ възел, алфа-стойността на който е по-голяма или равна на бета-стойността на съответния минимизиращ родител.
- *Оценка на алфа-бета процедурата:* При използване на алфа-бета процедурата се получава напълно точен резултат. Броят на генерираните и оценени възли при алфа-бета процедурата е най-малко от порядъка на $b^{d/2}$. Тази стойност се получава в идеалния случай, когато оценките на листата са наредени максимално добре. Следователно алфа-бета процедурата намалява скоростта на развитие на комбинаторния взрив, но не го предотвратява.



2.3. Намиране на цел при спазване на ограничителни условия (Задачи за удовлетворяване на ограничения).

Формулировка
Дадени са:

- **множество от променливи** $v_1, v_2, \dots, v_n$ (със съответни **области на допустимите стойности** Dv_i – дискретни (крайни или изброими безкрайни) или непрекъснати)
- **множество от ограничения** (допустими/недопустими комбинации от стойности на променливите)

Видове ограничения:

- **Унарни** - ограничения, които включват една променлива
- **Бинарни** - ограничения, които включват две променливи
- **От по висок ред** - ограничения, които включват три или повече променливи
- **Основани на предпочитания (Preferences или soft constraints)** - показват коя стойност е по-добра за дадена променлива.

Целево състояние (състояния): множество от свързвания със стойности на променливите $\{v_1=c_1, v_2=c_2, \dots, v_n=c_n\}$, които удовлетворяват всички ограничения.

** Целта на задачата (това, което се търси) е да се инициализират по такъв начин всички променливи със стойности от съответните им домейни, така че и същевременно всички ограничения да бъдат удовлетворени.*

Примерни задачи от разглеждания тип

- Задача за осемте (n-те) царици, криптоаритметика, оцветяване на географска карта, пъзели, кръстословици, судоку и др.
- планиране, проектиране, съставяне на разписания и др.

Алгоритми за решаване на задачи за търсене на цел при спазване на ограничителни условия:

- Генериране и тестване (generate and test)
- Търсене с възврат (backtracking)
  - *Евристика:* избор на най-малко ограничаващата променлива
- Разпространяване на ограниченията (constraint propagation)
  - Forward checking
  - Съвместимост на дъгите (Arc consistency)
- Локално търсещи алгоритми
  - MinConflicts (минимизиране на конфликтите)

** Предимство на задачите за удовлетворяване на ограниченията са, че първо са прост пример за формален език за представяне и второ позволява използването на полезни алгоритми с общо предназначение с повече мощност от стандартните алгоритми за търсене.*

3. Методи за информирано (евристично) търсене на път до определена цел.

Обща характеристика:
Реализират пълно изчерпване по гъвкава стратегия или търсене с отсичане на част от графа на състоянията. Приложими са при наличие на специфична информация за предметната област, позволяваща да се конструира оценяваща функция (евристика), която връща не булева оценка (числова оценка в предварително определен интервал). Тази оценка може да служи например за мярка на близостта на оценяваното състояние до целта или на необходимия ресурс за достигане от оценяваното състояние до целта. Най-често се използва евристична оценяваща функция *h*, която връща като резултат приблизителната стойност на определен ресурс, необходим за достигане от оценяваното състояние до целта.
** Добре е да се отбележи, че основното нещо, на което влияе една добра евристична функция е бързодействието на дадения алгоритъм. Но имайте предвид, че освен на него може да има положително влияние и на сложността по памет.*

Програмна реализация:
Чрез използване на работна памет (списък на т. нар. открити възли или списък от натрупани/изминати пътища, започващи от началния възел – фронт на търсенето). Възможните състояния, които следва да се обходят се сортират въз основа оценяваща функция, която казва кое състояние е по-добро (best-first - винаги вземаме най-доброто първо). Оценяваща функция може да е изцяло евристична (като при Greedy) или да в комбинация от евристична функция и така, която дава точна оценка (като при A*).
Примерна задача:

- Търсене на път между две селища (пътна карта на Румъния).

3.1. Метод на най-доброто спускане (търсене на най-добър път, Best-first search *(Greedy)*).

Сортиране на списъка в съответствие с евристичната функция (например h(node) = <оценка на цената на пътя от node до целта>).
Оценка на метода Best-first search:

- Методът е ефективен, но не е нито пълен (има опасност от зацикляне), нито оптимален.
- Времевата сложност на метода е $O(b^m)$, но използването на подходяща евристика може да доведе до съществено подобрение.
- Пространствената сложност на метода е $O(b^m)$, тъй като се съхраняват всички достигнати състояния.

3.2. Търсене с ограничена широчина (търсене в лъч, Beam search).

Ограничаване на списъка до първите ***l*** най-добри възела от него (в съответствие с евристиката). При ***l=1*** се доближава до метода на най-бързото изкачване.

Оценка на метода Beam search:

- Методът не е нито пълен, нито оптимален.
- Времевата сложност зависи от разгледаното “изрязано” пространство и евристиката. При използването на подходяща евристика може да доведе до съществено подобрение. Сложността е ***O(blm)***, където ***l*** е размера на ограничения списък.
- Пространствената сложност зависи от параметъра ***l***: ***O(bl)***.

3.3. Метод на най-бързото изкачване (Hill climbing).

Списъкът се ограничава до най-добрия му елемент (в съответствие с евристиката), и то само ако той е по-добър от своя родител. Търсенето е еднопосочно, без възможност за възврат.

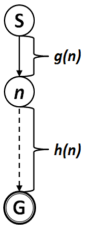
Оценка на метода Hill climbing:

- Методът не е нито пълен, нито оптимален.
- Времевата сложност е много малка - зависи от дълбочината на пространството и евристиката: ***O(bm)***.
- Пространствената сложност е константна: ***O(b)***.

Възможни проблеми:

- **Локален екстремум** – състоянието е по-добро от съседните (наследниците си), но не е най-доброто в цялото ПС;
- **Плато** – съседните състояния (наследниците на текущото състояние) изглеждат също толкова добри, колкото и текущото;
- **Хребет** – никой от възможните оператори не води до по-добро състояние от текущото, макар че два или повече последователни оператори биха могли да доведат до такова състояние.

3.4. Търсене с минимизиране на общата цена на пътя (A*).



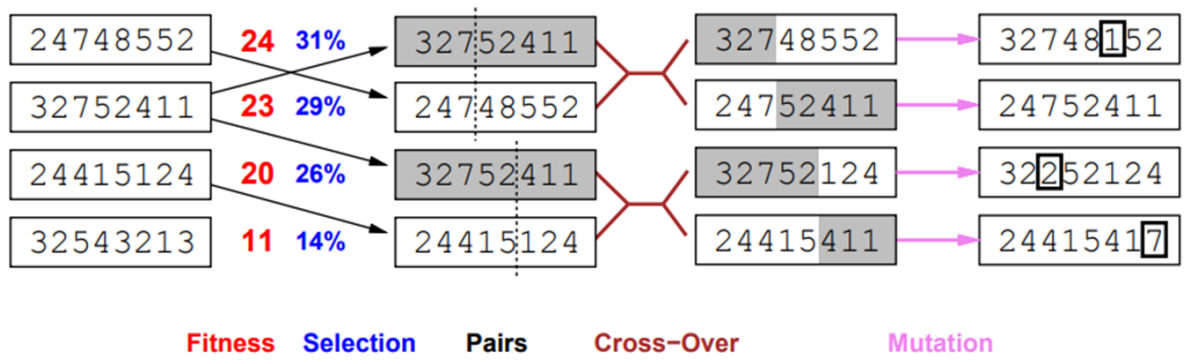
Комбинира търсене с равномерна цена на пътя с търсене на най-добър път. Списъкът се сортира в съответствие с функцията ***f(node) = g(node) + h(node)***. Тук функцията ***g*** връща като резултат цената на изминатия път от началния възел до ***node***, а евристичната функция ***h*** връща като резултат приблизителна стойност на цената на оставащата част от пътя от ***node*** до целта.

Оценка на метода A*:

- Стратегията е ***пълна***, ако разклоненията (наследниците) на всеки възел от ГС са краен брой и цената на преходите е положителна, и ***оптимална***, ако евристиката е приемлива (оптимистична), т.е. никога не надценява стойността (цената) ***h**** на оставащия път (ако ***h*(node) ≥ h(node)*** за всеки възел ***node***).
- Времевата и пространствената сложност на метода са експоненциални. Можем да кажем, че са приблизително $O(b^d)$, като зависят пряко от грешката на евристичната функция и дължината на най-късият път до решението.

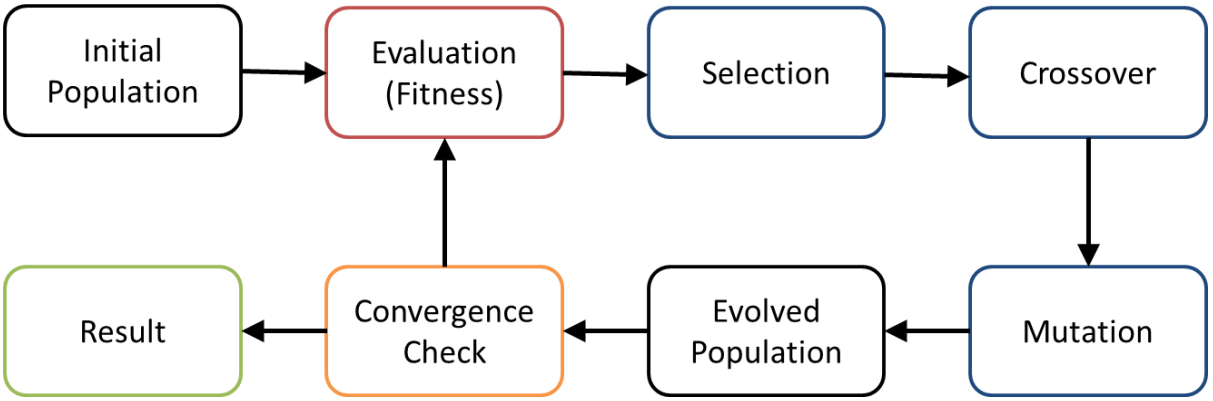
4. Генетични алгоритми - основен алгоритъм , типове кръстосване и мутация.

Дефиниция на генетичен алгоритъм:
Вариант на стохастично (локално) търсене в лъч, при което новите състояния се генерират чрез комбиниране на двойки родителски състояния вместо чрез модифициране на текущото състояние.



Основни принципи

- Състоянията се представят като низове над дадена крайна азбука (често като низове от нули и единици).
- Оценяваща функция (fitness function): оценява пригодността (близостта до целта) на съответното състояние. Има по-големи стойности за по-добрите състояния.
- Алгоритъмът започва работа с множество (популация) от k случайно генерирани състояния (поколение 0).
- Принципи на получаването на състоянията от следващите поколения: селекция, кръстосване, мутация.



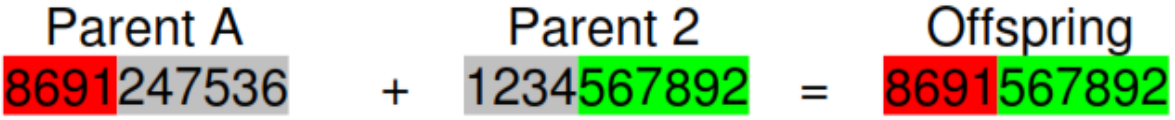
Селекция
В генерирането на състоянията от следващото поколение участват някои от най-добрите представители на текущото поколение (съгласно оценяващата функция), избрани на случаен принцип.

Кръстосване
Избират се двойка “родителски” състояния и се определя т. нар. точка на кръстосването им (позиция в двата низа). Състоянието – наследник се получава чрез конкатенация на началната част на първия и крайната част на втория родител. Възможно е да се получи и друг наследник, в конструирането на който участват неизползваните части на двамата родители.

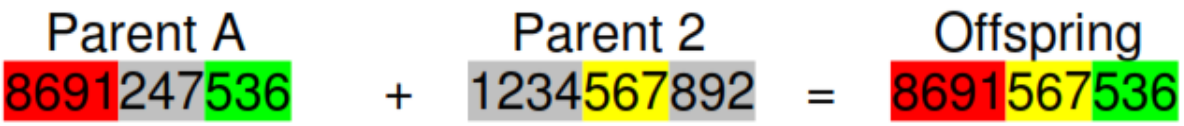
Мутация
Извършване на случайни промени в случайно избрана малка част от новата популация с цел да се осигури възможност за достигане на всяка точка от пространството на състоянията и да се избегне опасността от попадане в локален екстремум.

Примери за различни типове кръстосване

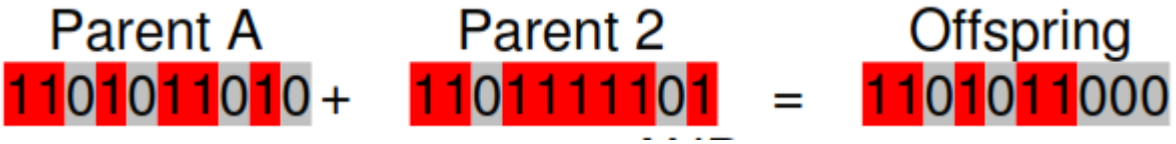
- **Кръстосване в единична точка:** Избира се една точка на кръстосване. Низът – резултат от началото си до точката на кръстосване е копие на началната част на единия родител, останалата му част е копие на съответната част на втория родител.



- **Кръстосване в две точки:** Избират се две точки на кръстосване. Низът – резултат от началото си до първата точка на кръстосване е копие на съответната част от първия родител, частта на резултата от първата точка на кръстосване до втората точка на кръстосване е копие на съответната част на втория родител и останалото е копие на оставащата след втората точка на кръстосване част на първия родител.



- **Аритметично кръстосване:** Извършва се определена операция (аритметична, логическа и т.н.) между двамата родители и в резултат се получава новото потомство.



На примера може да видите двоично кодиране, операция AND между съответните битове.

Примери за различни типове мутация

- Както и при кръстосването, така и при мутацията видът зависи от задачата, върху която ще се прилагат. Примерни варианти са:
  - Произволно променяне на знак от низа.
  - Размяна на два знака от низа.
  - Вмъкване на знак от низа на друга позиция (шифтва останалите).
  - Обръщане редът на знаците във вътрешен интервал.