Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informačné systémy

Prípadová štúdia:

N dám na šachovnici N×N

Predmet: ***Evolučné algoritmy*** Autor: ***Milan Skuhra***
Prednášateľ: ***Doc. RNDr. Jiří Pospíchal, DrSc.*** Dátum: ***30. jún 2006***

Obsah

Kódy umiestnenia dám na šachovnici

Vzhľadom na to, že na šachovnici $n \times n$ je vždy n dám, každá môže byť na samostatnom riadku a samostatnom stĺpci. Z tohto predpokladu budeme vychádzať, pretože to je zároveň podmienka preto aby sa dámy navzájom neohrozovali.
Dámy budeme na šachovnicu rozmiestňovať tak, že prvú dámu postavíme na prvý riadok do niektorého zo stĺpcov. Ďalšie dámy postupne ukladáme na riadky, pričom dámu nemožno umiestniť do stĺpca kde je už iná dáma. Poslednú dámu možno umiestiť už len na jedno miesto.

Základné kódovanie

Stav rozmiestnenia môžeme zapísať pomocou usporiadanej n–tice: $\{p_1, p_2 \mathbf{K}\, p_n\}$, kde $p \in \langle 1, n \rangle$. Pozícia v n–tici určuje riadok šachovnice a hodnota p_i stĺpec. Z hľadiska kombinatoriky ide o permutácie n prvkov bez opakovania a tak vieme určiť veľkosť priestoru $P(n) = n!$.

Logické kódovanie

Logické kódovanie lepšie odzrkadľuje skutočnosť – ako sa dámy na šachovnici rozmiestňujú. Je to tiež pole hodnôt, ale dĺžky $n - 1$. Pre jednoduchšie rozlíšenie budeme logické kódovanie uzatvárať do špicatých zátvoriek $\langle r_1, r_2 \mathbf{K}\, r_{n-1} \rangle$.

Interpretácia je nasledovná:

- Rovnako ako pri základnom kódovaní jednotlivé pozície postupnosti reprezentujú riadky.
- Posledný riadok sa nezapisuje, pretože tam už nemáme na výber, kam umiestniť poslednú dámu (už je iba jeden stĺpec voľný).
- Na každú pozíciu postupnosti zapíšeme poradové číslo stĺpca z tých stĺpcov na ktoré mohla byť dáma umiestnená. To znamená že pri určovaní poradových čísel preskočíme tie stĺpce, ktoré už sú obsadené.
- Na každej pozícií môže byť hodnota iba z určitého intervalu $r_i \in \langle 1, n - i + 1 \rangle$ kde $i \in \langle 1, n - 1 \rangle$

Transformácia zo základného na logické kódovanie

Vstupom transformácie je pole obsahujúce hodnoty $p_{i \in \langle 1,n \rangle}$ this.***data***. A výstupom pole s hodnotami $r_{i \in \langle 1,n-1 \rangle}$ ***result***.

```
var i, j, result = this.data.subArray(0, this.size - 1);
for(i = 1; i < result.length; i++)
      for(j = 0; j < i; j++)
            if(this.data[i] > this.data[j]) result[i]--;
return result;
```

Transformácia z logického na základné kódovanie

Vstupom je pole s hodnotami $r_{i \in \langle 1,n-1 \rangle}$ this.***data*** a výstupom pole s hodnotami $p_{i \in \langle 1,n \rangle}$ ***result***.

```
var i, j, result = this.data.clone(), aux = new Array(0);
result[result.length] = 1; // expanding array
for(i = 0; i < result.length; i++) {
      for(j = 0; j < i; j++) {
            if(aux[j] <= result[i]) result[i]++;
            else { aux = aux.insert(j, result[i]); break; }
      }
```

```
        if(aux.length == i) aux[i] = result[i];
}
return result;
```

Číselné kódovanie

Každé rozloženie dám na šachovnici možno vyjadriť číslom, budeme ho označovať písmenom *s*. Toto číslo zároveň reprezentuje poradie zo všetkých možných stavov, preto nadobúda hodnoty z intervalu $s \in \langle 0, n!-1 \rangle$.

Kódovanie *s* je možné transformovať z/na logické kódovanie pomocou vzťahu:

$$s = \sum_{i=1}^{n-1} (r_i - 1) \cdot (n - i)!$$

Toto kódovanie je úsporné, ľahko sa s ním pracuje (generovanie, mutovanie, kríženie), ale neumožňuje priamo zistiť či sa dámy na šachovnici ohrozujú (výpočet funkcie fitnes). Ďalšou nevýhodou sú nároky na rozsah typu čísla (napr. pre šachovnicu 32×32 je rozsah maximálna hodnota až po 2,631308369336935301672180121 6e+35).

Transformácia z logického na číselné kódovanie

Vstupom je $r_{i \in \langle 1, n-1 \rangle}$ v premennej this.***data*** a výstupom je hodnota *s* v premennej ***result***.

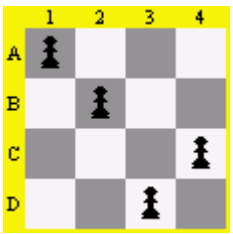
```
var i, offset = 1, base = 1, result = 0;
for(i = this.data.length - 1; i >= 0; i--) {
        result += (this.data[i] - 1) * base;
        base *= ++offset;
}
return result;
```

Transformácia z číselného na logické kódovanie

Vstupom je *s* v premennej this.***data*** a výstupom $r_{i \in \langle 1, n-1 \rangle}$ v premennej ***result***.

```
var size = this.size, nm = this.data;
var i, result = new Array(this.size - 1), base = 1, aux = --size;
while(size > 1) base *= size--; //factorial: (size - 1)!
for(i = 0; i < result.length; i++) {
        result[i] = Math.floor(nm / base) + 1;
        nm %= base;
        base /= aux--;
}
return result;
```

Príklad kódovania pre n = 4



Kódovanie	Hodnota
Základné	{1,2,4,3}
Logické	<1,1,2>
Číselné	1

Ohodnotenie fitnes

Fitnes jedinca je odvodené od počtu ohrození všetkých dám na šachovnici navzájom. Minimálny počet ohrození je nula a maximálny počet je určený výrazom:

$$o_{\max} = \frac{(n-1).n}{2}$$

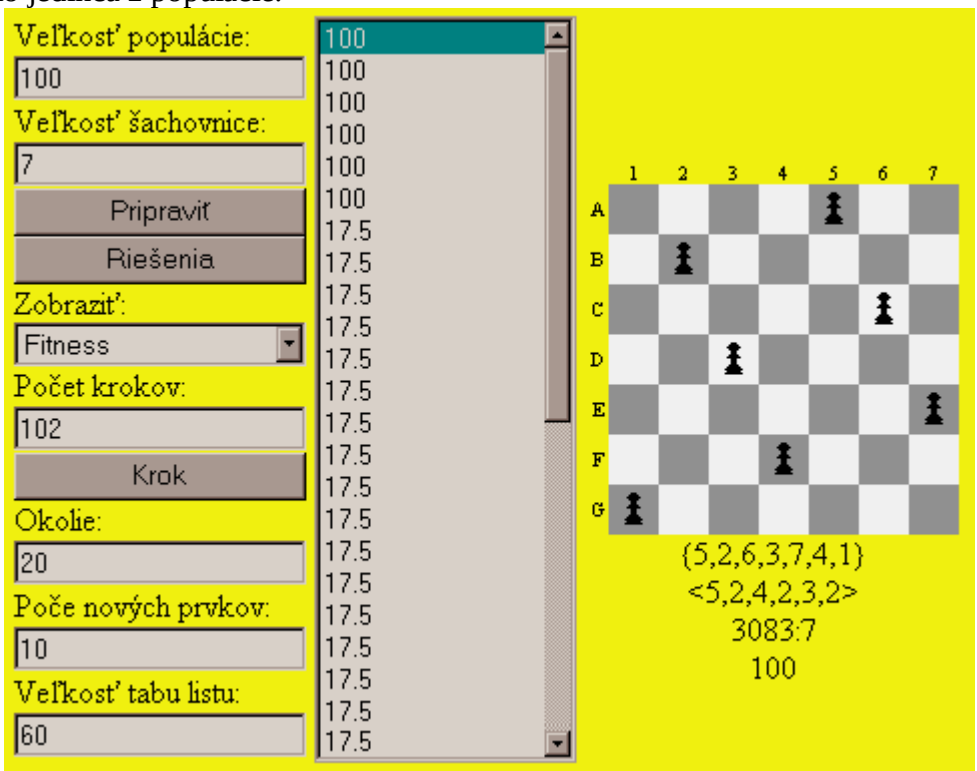
Kde n je rozmer šachovnice $n \times n$. Pokiaľ je $o = o_{\max}$, každá dáma ohrozuje všetky ostatné (keď sa nachádzajú na diagonále).
Teda definičný obor pre funkciu fitnes je $o \in \langle 0, o_{\max} \rangle$. Obor hodnôt je normovaný tak, aby bol z intervalu $f \in \langle 1{,}100 \rangle$. Samotná funkcia pre výpočet fitnes je:

$$f = \frac{100 \cdot o_{\max}}{o_{\max} + 99 \cdot o}$$

Algoritmus na zistenie počtu ohrození dám na šachovnici funguje nad základným kódovaním:

Implementácia

Hlavná časť implementácie bola riešená v jazyku JScript 5.6 a grafické časti pomocou DHTML. Produkt je spustiteľný v Internet Explorer 6.0.
Rozhranie je na Obr. 1. V ľavej časti sú ovládacie prvky, v strednej a v pravej zobrazenie vybraného jedinca z populácie.

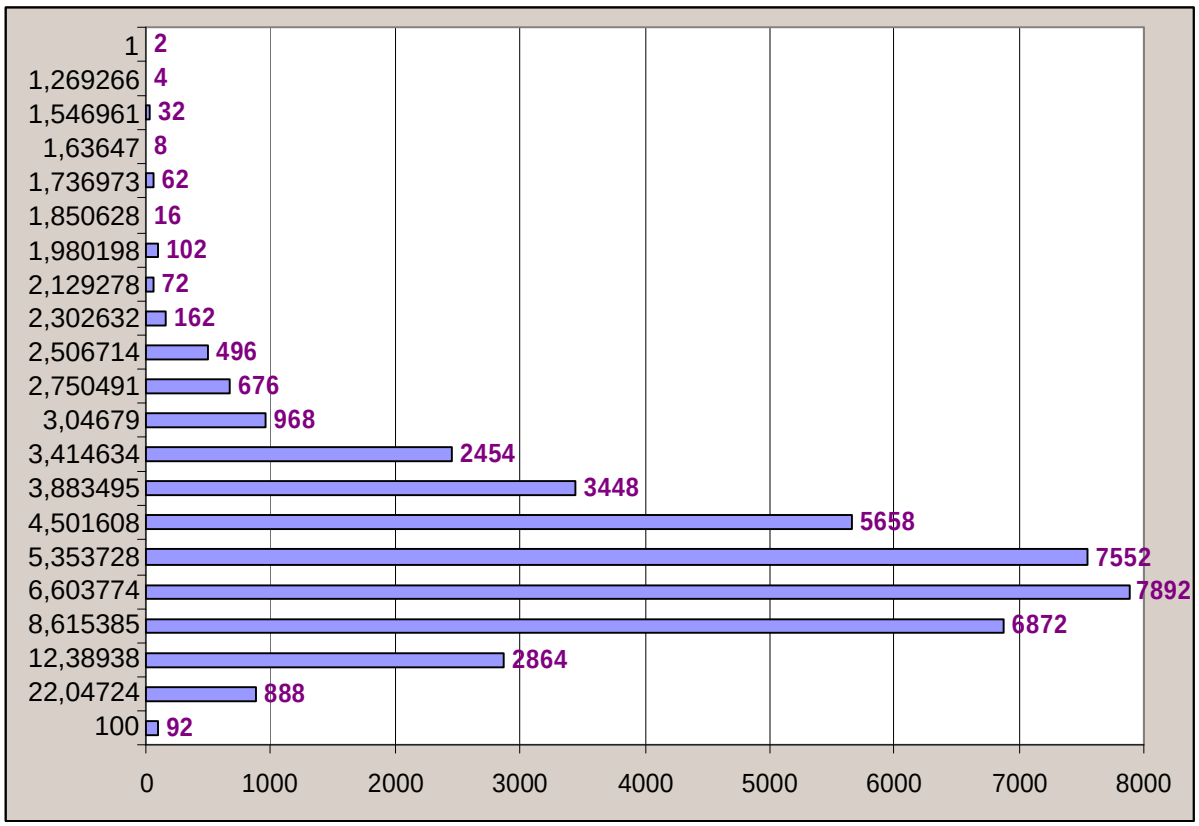


Obr. 1 Používateľské rozhranie

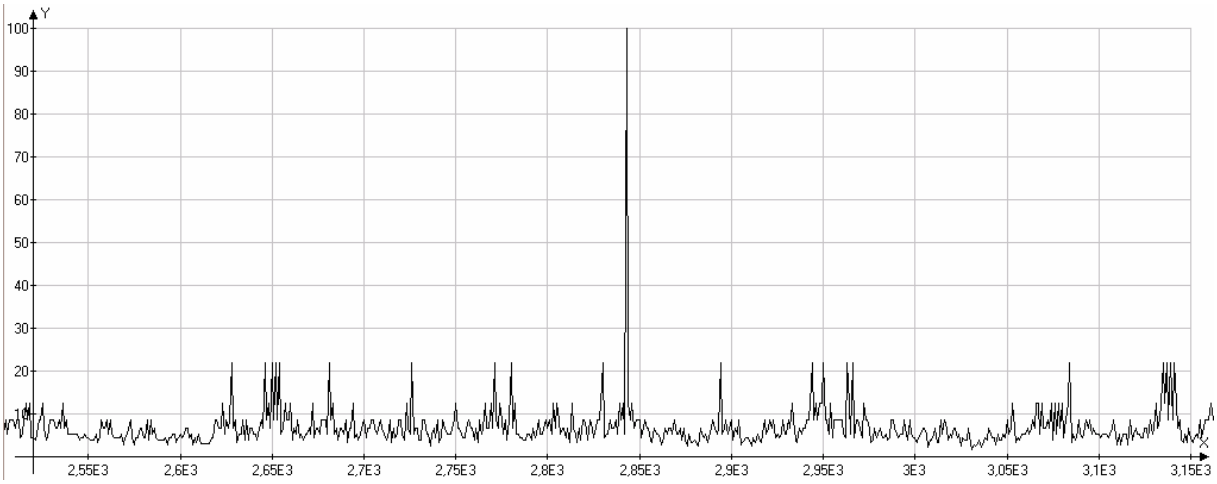
- **Veľkosť populácie** – nastavuje maximálnu veľkosť populácie, ktorá by mala byť menšia vždy väčšia ako *Veľkosť tabu listu*. Po stlačení tlačidla *Riešenie* sa zmení hodnota tohto poľa na počet nájdených riešení.
- **Veľkosť šachovnice** – je potrebné nastaviť pred stlačením tlačidla *Pripraviť* alebo tlačidla *Riešenia*.
- **Pripraviť** – tlačidlom sa inicializuje populácia o veľkosti nastavenej v *Veľkosť populácie* s náhodne vygenerovanými jedincami.
- **Riešenia** – použitie hrubej sily na prehľadanie celého stavového priestoru a nájdenie všetkých riešení. Neodporúčam skúšať pre šachovnice väčšie ako 10x10. Riešenia vloží do populácie a nastaví pole *Veľkosť populácie*, ktorú v tomto prípade znamená počet riešení.
- **Zobraziť** – umožňuje prepínať medzi zobrazeniami jedincov v populácia. Na výber sú tri druhy kódovania a vyhodnotenie funkcie fitnes.

- **Počet krokov** – ukazuje počet krokov, ktoré boli od inicializovanej populácie (tlačidlo *Pripraviť*) vykonané. Jeden krok spočíva vo výbere vhodného jedinca (*Veľkosť tabu listu*), vygenerovaní nových jedincov (*Počet nových prvkov*) z určitého okolia (*Okolie*) a následne sú tieto nové jedince vložené do populácie. Populácia sa usporiada podľa hodnoty fitnes a skráti o nadbytočné jedince (*Veľkosť populácie*). Pokiaľ do tohto políčka zapíšeme číslo väčšie ako je aktuálne – možno vykonať viac krokov naraz.
- **Krok** – tlačidlo na vykonanie jedného príp. viacerých krokov horolezeckého algoritmu.
- **Okolie** – určuje okolie v akom sa generujú náhodné jedince vzhľadom na vybraného jedinca v danom kroku. Najlepšie sa to dá predstaviť pomocou číselného kódovania, kde okolie bodu s je z množiny $<-o,0) \cup (0,o>$ pričom o je hodnota v políčku *Okolie*.
- **Počet nových prvkov** – určuje koľko nových jedincov sa vygeneruje v jednom kroku.
- **Veľkosť tabu listu** – určuje na koľko krokov je prvok, ktorý už bol vybraný, vylúčený z výberu.

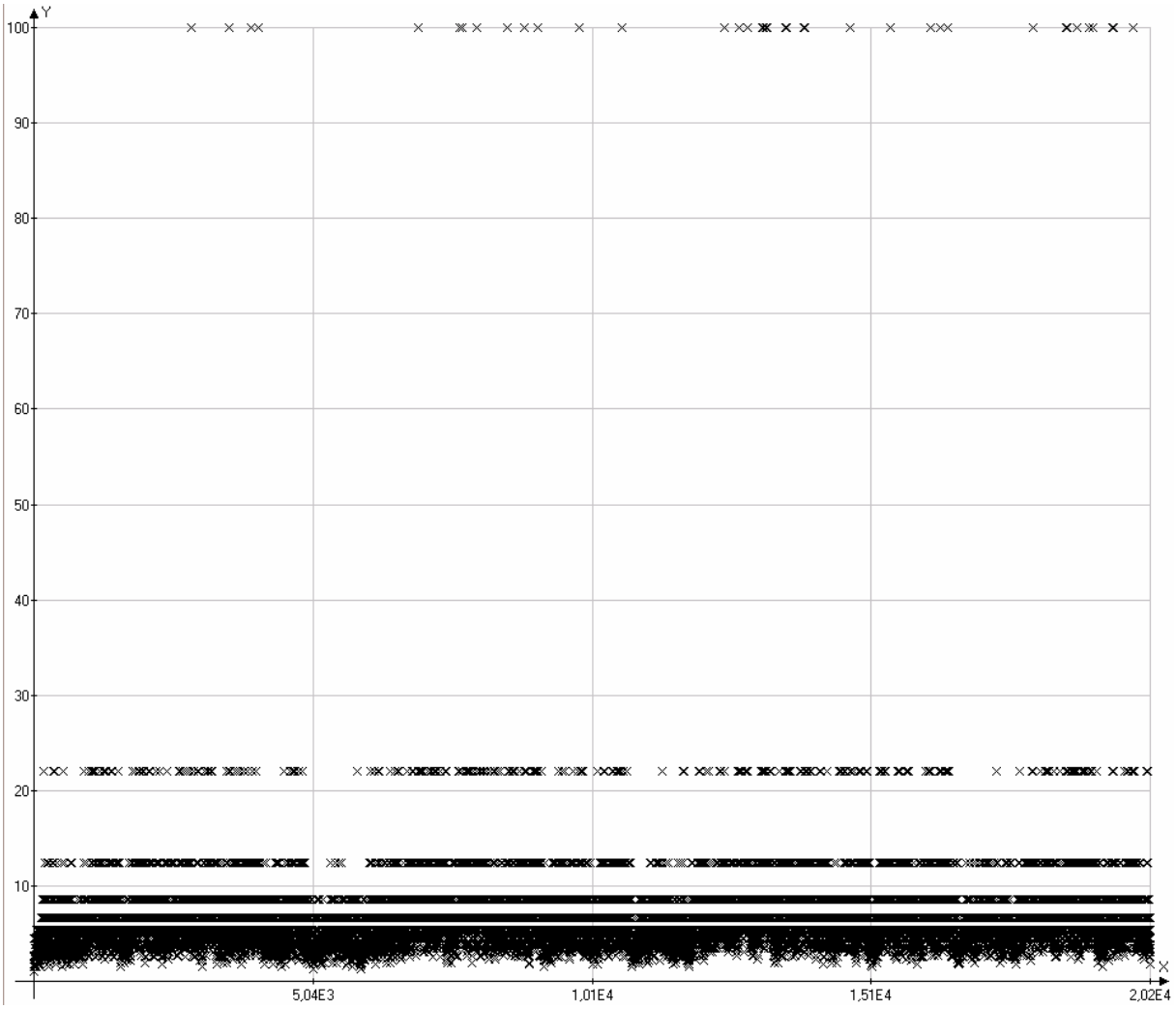
Štatistické údaje



Početnosť jedincov v celom stavovom priestore vzhľadom na hodnotu funkcie fitnes pre N=8



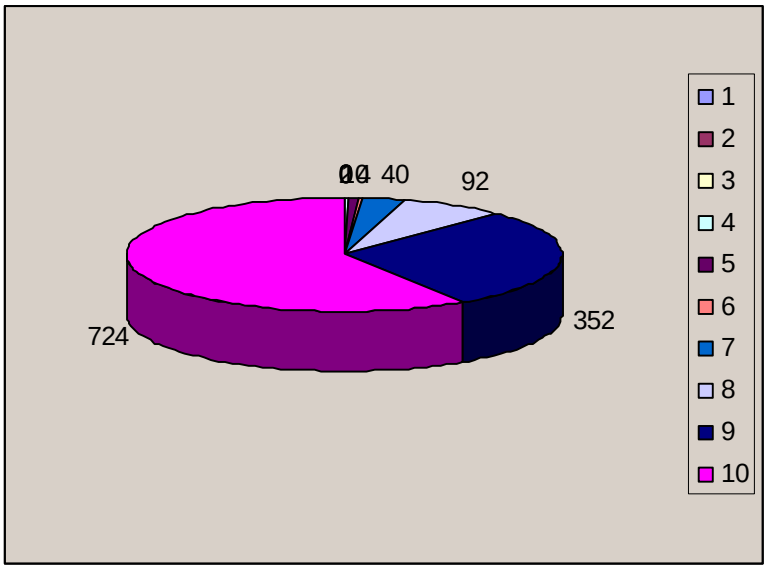
Výsek funkcie fitnes pre ilustráciu jej vlastnosti.



Polovica stavového priestoru (druhá je symetrická) s funkčnými bodmi funkcie fitnes pre ilustráciu ne spojitosti.

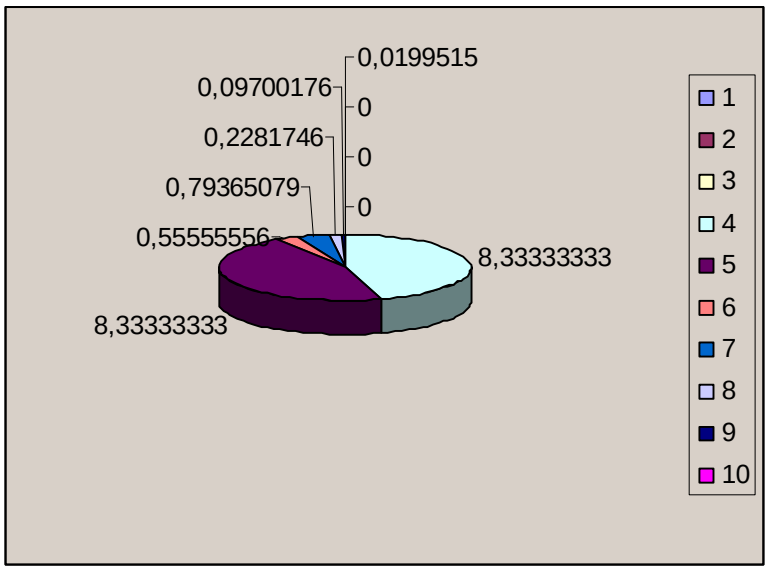
N	#riešenia
1	0
2	0
3	0
4	2
5	10
6	4
7	40
8	92
9	352
10	724

Počet riešení pre veľkostí šachovníc 1 až 10.



N	p()
1	0
2	0
3	0
4	8,33333333
5	8,33333333
6	0,55555556
7	0,79365079
8	0,2281746
9	0,09700176
10	0,0199515

Pravdepodobnosť nájdenie riešenia pomocou slepého algoritmu pre veľkosti šachovníc 1 až 10.



Záver

Práca sa zaoberala problémom rozmiestnenia N dám na šachovnici NxN. Boli navrhnuté tri druhy možného kódovania jedného zo stavov a konverzie medzi nimi. Každé kódovanie má určité výhody a určité nevýhody, ktoré sú popísané v práci. Na overenie riešenia bol použitý jazyk JScript, v ktorom sa dali pomerne jednoducho vyjadriť aj zložité algoritmy. Táto implementácia však prakticky neumožňuje hľadať riešenia na šachovnici pre N > 9.
Boli implementované dve metódy na hľadanie riešenia. Prehľadávaním celého priestoru a pomocou horolezeckého algoritmu zo zakázaným prehľadávaním. Obe metódy funguje spoľahlivo pre N > 9.

Použitá literatúra

[1] V.Kvasnička, J.Pospíchal, P.Tiňo: Evolučné algoritmy, STU Bratislava, 2000, ISBN: 80-227-1377-5.

[2] P.Návrat a Kol.: Umelá Inteligencia, STU Bratislava, 2002., ISBN 80-227-1645-6.

[3] Emily A. Vander Veer: JavaScript For Dummies, Computers 3th Edition October 2000, ISBN: 0764506331