# Riešenie rozvrhovacieho problému pre procesory pomocou genetického algoritmu

Alexander Moravčík
7moravcik@st.fmph.uniba.sk

## 1. Úvod

Určenie optimálneho rozvrhu pre danú množinu úloh je problém značného teoretického i praktického významu. Cieľom je minimalizovať množstvo použitých prostriedkov (čas, peniaze, energia, ľudské zdroje), nutných na realizáciu týchto úloh. Nanešťastie, užitočnosť algoritmu poskytujúceho optimálne riešenie tohto problému je vážne naštrbená jeho výpočtovou náročnosťou – ako sa totiž ukázalo, ide o NP-úplný problém. K slovu sa tak dostávajú rozličné techniky poskytujúce aspoň suboptimálne riešenie, avšak v rozumnom čase: od jednoduchých greedy algoritmov až po simulované žíhanie, neurónové siete, evolučné stratégie alebo genetické algoritmy. Práve použitie posledne menovanej metódy pri riešení špeciálneho prípadu rozvrhovacieho problému je témou tejto práce.

## 2. Definícia problému

Nech $P = \{p_0, \ldots, p_m\}$ je množina procesorov a $C = \{c_1, \ldots, c_n\}$ množina komunikačných kanálov, ktorými sú procesory navzájom prepojené, pričom $m \geq 0$, $n > 0$ (ak $m = 0$, môže byť prípadne $C = \emptyset$). Nech ďalej $S = \{s_1, \ldots, s_k\}$, $k \geq 1$ je množina úloh, ktoré treba na procesoroch vykonať. Každá úloha $s_i$ je definovaná ako usporiadaná dvojica kladných celých čísel $s_i = \left(t^i_{comp}, t^i_{comm}\right)$, kde $t^i_{comp}$ je čas potrebný na výpočet úlohy a $t^i_{comm}$ čas potrebný na prenos vypočítaných výsledkov na hlavný procesor $p_0$. Úlohy vykonané na hlavnom procesore, pochopiteľne, nespotrebujú žiaden komunikačný čas. Procesory $p_1, \ldots, p_m$ zdieľajú kanály $C$, nie však paralelne, čiže nie je možné, aby jedným kanálom dva procesory posielali výsledky v rovnakom čase. Úlohy sú nepreemptívne, tj. úlohu, ktorá sa už začala vykonávať nie je možné prerušiť. Výnimkou je vynútené prerušenie v prípade, že procesor dokončil výpočtovú časť úlohy, no žiaden komunikačný kanál nie je práve voľný. Vtedy musí čakať, pokým sa nejaký neuvoľní. Ak dva procesory žiadajú o pridelenie uvoľneného kanálu v rovnakom čase, prednosť má ten s nižším indexom.

Cieľom je nájsť rozvrh minimalizujúci celkový čas nutný k vykonaniu množiny úloh $S$ na systéme pozostávajúcom z procesorov $P$ a komunikačných kanálov $C$. Formálnejšie, rozvrhom je dvojica $\sigma = (\pi, \omega)$, kde $\pi : P \to 2^S$ je zobrazenie, ktoré každému procesoru priradí množinu úloh, ktoré sa na ňom majú vykonať (pričom $\bigcup_{p \in P} \pi(p) = S$) a $\omega = \{\prec_p \mid p \in P\}$ je množina totálnych usporiadaní na množinách $\pi(p), p \in P$, určujúce poradie vykonávania úloh na jednotlivých procesoroch. Hľadáme taký rozvrh $\sigma_{opt}$, pre ktorý platí

$$t_{exec}(\sigma_{opt}) = \arg\min_{\sigma} t_{exec}(\sigma).$$

## 3. Genetický algoritmus

Genetický algoritmus je v podstate akousi šablónou na riešenie takmer ľubovolného problému. Z tejto jeho všeobecnosti vyplýva, že pri konštrukcii algoritmu riešiaceho náš konkrétny rozvrhovací problém, treba genetický algoritmus „nainštancovať“. V nasledujúcich odstavcoch budú postupne popísané zvolené relevantné dátové štruktúry, funkcie, operátory a mechanizmy.

### 3.1. Reprezentácia genotypu

Genotyp je reprezentovaný priamo, tj. chromozóm nie je reťazec (binárnych) symbolov ale priamo samotný rozvrh. Presnejšie, rozvrh, tak ako bol definovaný v časti 2 a chromozóm sú navzájom jednoducho transformovateľné. Chromozómom je dvojica $\bar{\sigma} = (\bar{\pi}, \bar{\omega})$, kde $\bar{\pi} \in \{0,\ldots,m\}^k$ má v $i$-tej súradnici číslo procesora, na ktorom sa bude vykonávať $i$-tá úloha a $\bar{\omega} \in Perm(\{1,\ldots,k\})$ je permutáciou množiny indexov úloh, interpretovaná ako totálne usporiadanie na množine všetkých úloh. Pre rozvrh $\sigma = (\pi, \omega)$, ktorý zodpovedá chromozómu $\bar{\sigma} = (\bar{\pi}, \bar{\omega})$ potom platí $\pi(p) = \{s_i \in S \mid \bar{\pi}(i) = p\}$ a takisto $\omega$, množina usporiadaní podmnožín množiny $S$, je jednoznačne určená usporiadaním celej $S$.

Priamym dôsledkom použitia priamej reprezentácie je fakt, že operácie ako napr. kríženie, mutovanie alebo generovanie náhodného chromozómu sú problémovo závislé, a teda nemožno použiť klasické generické procedúry, ktoré by ich realizovali.

### 3.2. Účelová funkcia

Pre daný chromozóm $\bar{\sigma}$ vracia účelová funkcia $f$ množstvo časových jednotiek, potrebných k vykonaniu rozvrhu $\sigma$ s týmto chromozómom združeným pri známej hardvérovej konfigurácii multiprocesorového systému: $f(\bar{\sigma}) = t_{exec}(\sigma)$. Nejde teda o aproximáciu, ale o presný čas, ktorý uplynie od započatia prvého výpočtu po koniec poslednej komunikácie. Algoritmus, ktorý počíta hodnotu účelovej funkcie pracuje približne tak, že si v prvom kroku zistí, kedy nastane nasledujúca udalosť – koniec výpočtu alebo komunikácie – a v druhom kroku sa posunie do tohto časového okamihu a patrične zmodifikuje stav simulovaného systému. Tieto dva kroky sa opakujú dovtedy, kým sa nedokončí posledná úloha.

### 3.3. Fitness

Fitness chromozómu $\bar{\sigma}$ vrámci populácie $\bar{\Sigma}$ je definovaná ako

$$F(\bar{\sigma}) = \frac{0{,}99 f(\bar{\sigma}) + 0{,}01 f_{\min} - f_{\max}}{f_{\min} - f_{\max}},$$

kde $f_{\min} = \min_{\bar{\sigma} \in \bar{\Sigma}} f(\bar{\sigma})$ a $f_{\max} = \max_{\bar{\sigma} \in \bar{\Sigma}} f(\bar{\sigma})$. Tento tvar funkcie vznikol lineárnym zobrazením funkčných hodnôt účelovej funkcie na fitness, ktoré minimálnej (resp. maximálnej) funkčnej hodnote priraďuje maximálnu (resp. minimálnu) fitness $F_{\max} = 1$ (resp. $F_{\min} = 0{,}01$). Ak $f_{\min} = f_{\max}$, potom je populácia homogénna vzhľadom na hodnotu účelovej funkcie, a preto môžeme položiť $F(\bar{\sigma}) = 1$ pre všetky $\bar{\sigma} \in \bar{\Sigma}$.

### 3.4. Selekčný mechanizmus

Kvázináhodný výber chromozómov z populácie sa realizuje pomocou rulety na základe renormalizovaných fitness. Týmto spôsobom sa najprv vyberú dva chromozómy z rodičovskej populácie. Ak náhodne vygenerované číslo z intervalu $[0,1)$ je menšie ako pravdepodobnosť reprodukcie $P_{repro}$, potom sa tieto chromozómy skrížia a výsledné chromozómy sa po zmutovaní pridajú do priebežne vytváranej populácie potomkov. V opačnom prípade sa do tejto populácie priamo skopírujú. Toto sa opakuje dovtedy, pokým počet jedincov (zo začiatku prázdnej) následníckej generácie nedosiahne počet jedincov v pôvodnej populácii.

V prípade zapojenia elitizmu sa hneď na začiatku procesu vytvárania nasledujúcej generácie do nej prenesie najlepšie riešenie z aktuálnej populácie. Zabráni sa tak tomu, aby mohlo byť „zabudnuté“ kvázináhodným výberom rulety. Vedľajším efektom je to, „vektor prehľadávania“ stavového priestoru bude v nasledujúcej generácii nasmerovaný viac k tomuto riešeniu.

### 3.5. Operátor kríženia

Pri krížení chromozómov $\bar{\sigma}_1 = (\bar{\pi}_1, \bar{\omega}_1)$ a $\bar{\sigma}_2 = (\bar{\pi}_2, \bar{\omega}_2)$ sa jednotlivé podčasti krížia separátne, tj. štruktúra operátoru kríženia $O_{cross}$ vyzerá nasledovne:

$$O_{cross}(\bar{\sigma}_1, \bar{\sigma}_2) = \left(O'_{cross}(\bar{\pi}_1, \bar{\pi}_2), O''_{cross}(\bar{\omega}_1, \bar{\omega}_2)\right).$$

Kríženie je viacbodové, čiže to, či bude nejaká súradnica vektorov $\bar{\pi}_1, \bar{\pi}_2$ (resp. $\bar{\omega}_1, \bar{\omega}_2$) bodom kríženia sa rozhoduje pre každú z nich zvlášť, a to s pravdepodobnosťou $P'_{cross}$ (resp. $P''_{cross}$). V prípade $O'_{cross}$ sa prislúchajúce časti jednoducho vymenia, zatiaľ čo v prípade $O''_{cross}$ je po výmene nutné použiť opravný algoritmus (partial matching).

### 3.6. Operátor mutácie

Operátor mutácie $O_{mut}$, podobne ako operátor kríženia, pracuje odlišne na jednotlivých podčastiach chromozómu:

$$O_{mut}((\bar{\pi}, \bar{\omega})) = \left(O'_{mut}(\bar{\pi}), O''_{mut}(\bar{\omega})\right).$$

Operátor mutuje viacbodovo, každú súradnicu $\bar{\pi}$ (resp. $\bar{\omega}$) s pravdepodobnosťou $P'_{mut}$ (resp. $P''_{mut}$). V prvom prípade mutácia súradnice znamená nahradenie hodnoty v nej náhodným číslom procesora. V druhom prípade zasa výmenu s hodnotou v náhodne vybranej súradnici, pričom proces mutácií súradníc je rekurentný.

### 3.7. Generátor náhodného chromozómu

Náhodný chromozóm $(\bar{\pi}, \bar{\omega})$ sa vygeneruje tak, že hodnotám v súradniciach $\bar{\pi}$ sa priradia náhodné čísla procesorov a $\bar{\omega}$ sa priradí náhodná permutácia indexovej množiny úloh.

## 4. Testovacia sada

Testovacia sada úloh $S_{test}$, použitá pri nasledujúcich experimentoch, pozostáva z dvadsiatich úloh ($k = 20$) a zapísaná vo forme vstupu priloženého programu vyzerá nasledovne:

```
( 7 16) (11 22) (12 40) (15 22) (17 23)
(17 23) (19 23) (20 28) (20 27) (26 27)
(28 31) (36 37) (31 29) (28 22) (23 19)
(22 18) (22 17) (29 16) (27 16) (35 15)
```

Podľa M. D. KIDWELLOVEJ[1] je minimálny čas, potrebný na vykonanie tejto množiny úloh na troch procesoroch s jedným komunikačným kanálom ($m = 2$, $n = 1$) 202 časových krokov ($t_{exec}(\sigma_{opt}) = 202$).

## 5. Experimentálne výsledky

Grafy 1–8 zachytávajú úspešnosť genetického algoritmu pri hľadaní optimálneho riešenia rozvrhovacieho problému pre úlohy $S_{test}$, definované v predchádzajúcom odseku. Na horizontálnej osi sú vynesené poradové čísla generácií, zvislá os zachytáva kvalitu tej-ktorej generácie rozvrhov pri troch uhloch

[1] KIDWELL, M. D.: Using genetic algorithms to schedule distributed tasks on a bus-based system. In: *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA 93)*. Urbana-Champaign, IL, 1993. s. 368–374.

pohľadu, ktoré sú dané funkciami $e_1$, $e_2$ a $e_3$. Označme $\overline{\Sigma}^i$ $i$-tú generáciu chromozómov a $\overline{\sigma}^i_{elite} \in \overline{\Sigma}^i$ nech je taký chromozóm, aby platilo $t_{exec}(\sigma^i_{elite}) = \arg\min_{\overline{\sigma} \in \overline{\Sigma}^i} t_{exec}(\sigma)$, kde $\sigma^i_{elite}$ (resp. $\sigma$) je rozvrh združený s chromozómom $\overline{\sigma}^i_{elite}$ (resp $\overline{\sigma}$). Potom pre $e_1$, reprezentovanú hrubou čiernou krivkou, platí

$$e_1(i) = t_{exec}(\sigma^i_{elite}) \text{ pri aktivovanom elitizme;}$$

pre $e_2$ (tenká čierna krivka) platí

$$e_2(i) = \arg\min_{j \le i} t_{exec}(\sigma^j_{elite}) \text{ bez elitizmu;}$$

a nakoniec $e_3$ (tenká šedá krivka) vyzerá nasledovne:

$$e_3(i) = t_{exec}(\sigma^i_{elite}), \text{ takisto bez elitizmu.}$$

Pre zjednodušenie porovnávania sú hodnoty funkcií $e_1$ až $e_3$ na všetkých grafoch zobrazené pre generácie 0 až 10 000 a spriemerované na intervaloch širokých 100 generácií pri 10 zbehnutiach algoritmu od nultej (náhodnej) generácie. Hodnoty na zvislej osi sú ohraničené zdola 210 a zhora 280 časovými jednotkami.

Kľúčové parametre genetického algoritmu boli zvolené nasledovne:

$$P_{repro} = 0{,}5;\ P'_{cross} = 0{,}05;\ P''_{cross} = 0{,}025;\ P'_{mut} = 0{,}05;\ P''_{mut} = 0{,}05.$$

Tieto hodnoty sa počas experimentovania ukázali ako najvýhodnejšie. Veľkosť populácie $|\overline{\Sigma}|$ sa pohybuje v rozmedzí 2 (graf 1) až 500 (graf 8).

## 6. Záver

Celkovo najlepší dosiahnutý rozvrh pre sadu úloh $S_{test}$, prezentovaný ako výstup riešiaceho programu, vyzerá nasledovne:

```
( 7 16)  (p0    ( 81  88)          )
(11 22)  (p0    (116 127)          )
(12 40)  (p0    (127 139)          )
(15 22)  (p1 c1 ( 39  54) ( 54  76))
(17 23)  (p0    (159 176)          )
(17 23)  (p0    (176 193)          )
(19 23)  (p0    ( 36  55)          )
(20 28)  (p0    (139 159)          )
(20 27)  (p0    (193 213)          )
(26 27)  (p0    ( 55  81)          )
(28 31)  (p0    ( 88 116)          )
(36 37)  (p0    (  0  36)          )
(31 29)  (p2 c1 (137 168) (168 197))
(28 22)  (p1 c1 (118 146) (146 168))
(23 19)  (p1 c1 ( 76  99) ( 99 118))
(22 18)  (p2 c1 ( 54  76) ( 76  94))
(22 17)  (p1 c1 (  0  22) ( 22  39))
(29 16)  (p1 c1 (168 197) (197 213))
(27 16)  (p2 c1 ( 94 121) (121 137))
(35 15)  (p2 c1 (  0  35) ( 39  54))
```

Každý riadok predstavuje rozvrh pre úlohu, ktorej výpočtový a komunikačný čas sa nachádza v najľavejšej zátvorke. V zátvorke napravo nasledujú: pridelený procesor, komunikačný kanál, ktorým sa budú prenášať

výsledky na hlavný procesor, čas začatia výpočtu, čas ukončenia výpočtu, čas začiatku komunikácie a čas skončenia komunikácie. V prípade, že sa úloha vykonáva na hlavnom procesore $p_0$, komunikačný kanál a príslušné komunikačné časy zo zjavných príčin absentujú.

Bohužiaľ, čas potrebný k vykonanie úloh podľa tohto rozvrhu je 213 časových krokov, čo nekorešponduje so známou optimálnou hodnotou 202. Navyše, už len získanie niektorého z rozvrhov z množiny $\{\sigma \mid t_{exec}(\sigma) = 213\}$ sa ukázalo byť dosť namáhavé a nepravdepodobné. Nepomáhali žiadne zmeny parametrov počas behu programu, dokonca ani takých, ktoré si vyžadovali zmenu v kóde (zmena fitness, modifikácia pravdepodobností počas výpočtu, apod.).

Potvrdila sa však všeobecne známa skutočnosť, že genetický algoritmus je aj v „minimalistickej“ verzii schopný nájsť relatívne dobré riešenie zložitého optimalizačného problému v mimoriadne krátkom čase.

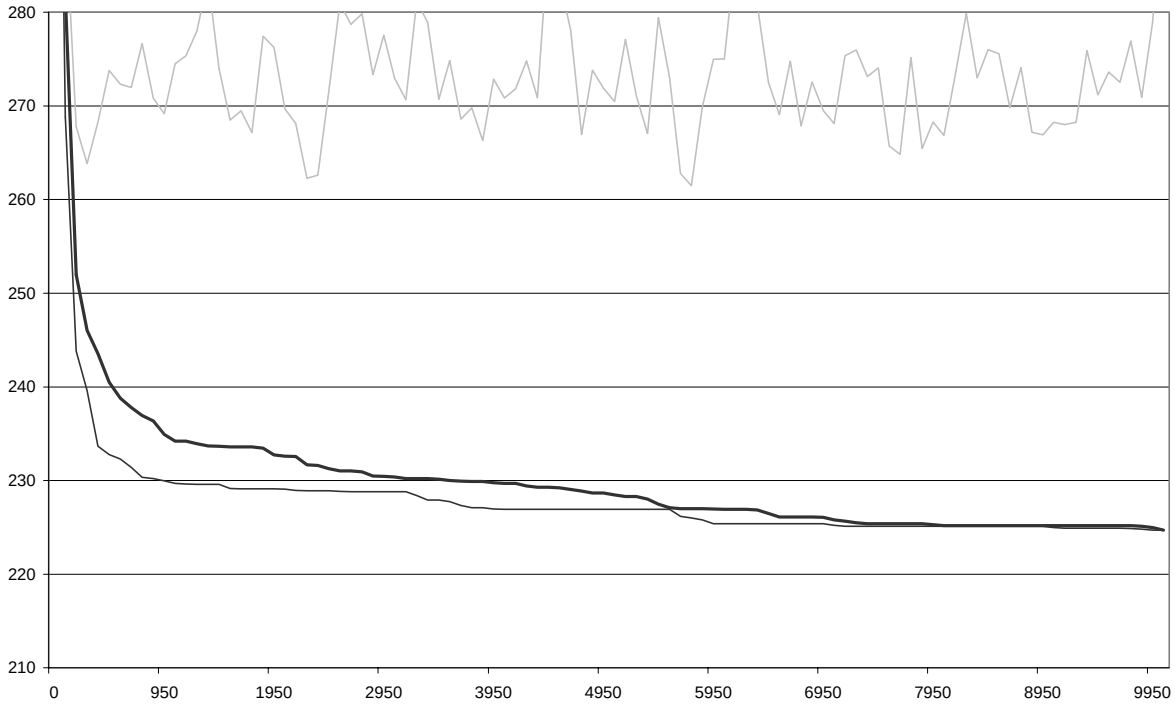
## 7. Použitá literatúra


[1] DUSSA-ZIEGER, K., SCHWEHM, M.: *Scheduling of parallel programs on configurable multiprocessors by genetic algorithms*. 1998.

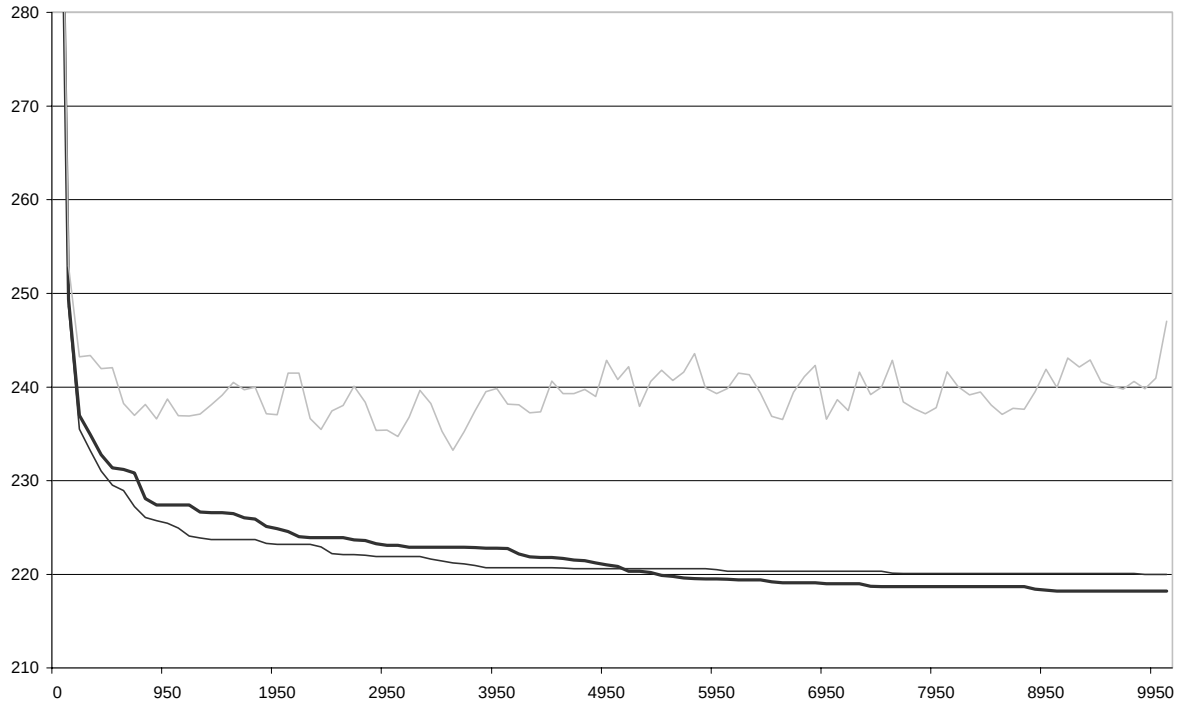
[2] KVASNIČKA, V., POSPÍCHAL, J., TIŇO, P.: *Evolučné algoritmy*. Bratislava: Vydavateľstvo STU, 2000. ISBN 80-227-1377-5

[3] REBREYEND, P., SANDNES, F. E., MEGSON, G. M.: *Static multiprocessor task graph scheduling in the genetic paradigm: A comparison of genotype representations*. 1998.

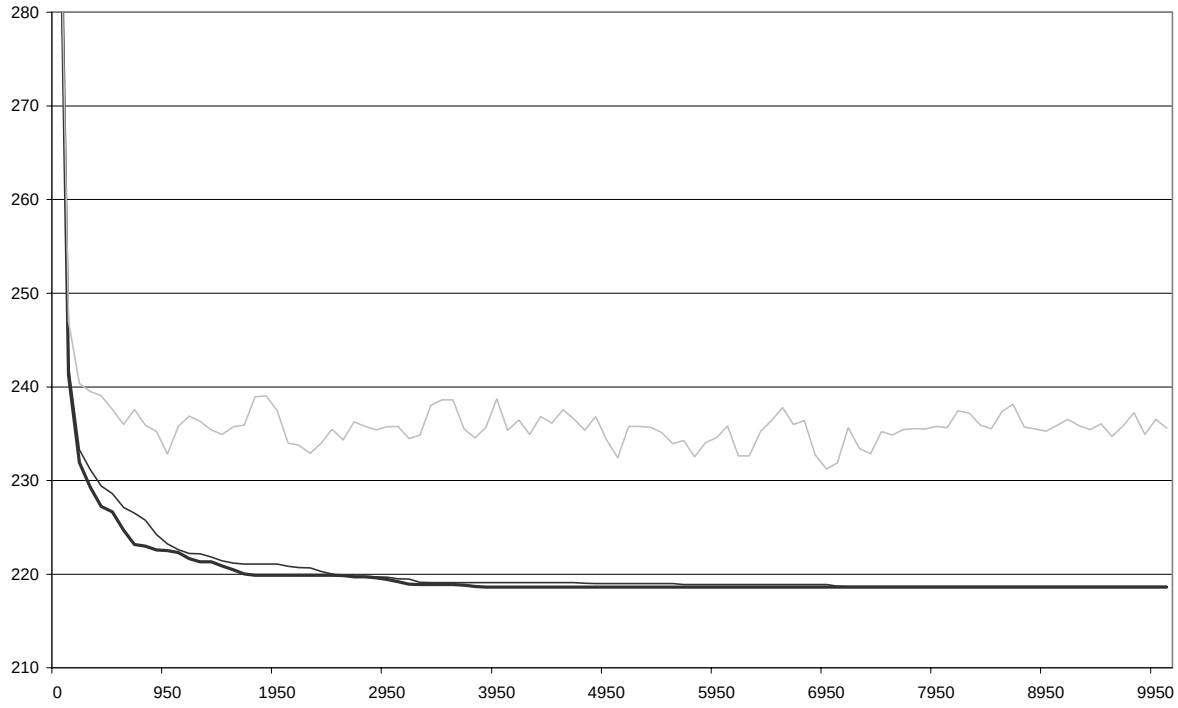
[4] PICO, C. A. G., WAINWRIGHT, R. L.: Dynamic scheduling of computer tasks using genetic algorithms. In: *Proceedings of the First IEEE Conference on Evolutionary Computation*. Orlando, Florida, 1994. s. 829–833.



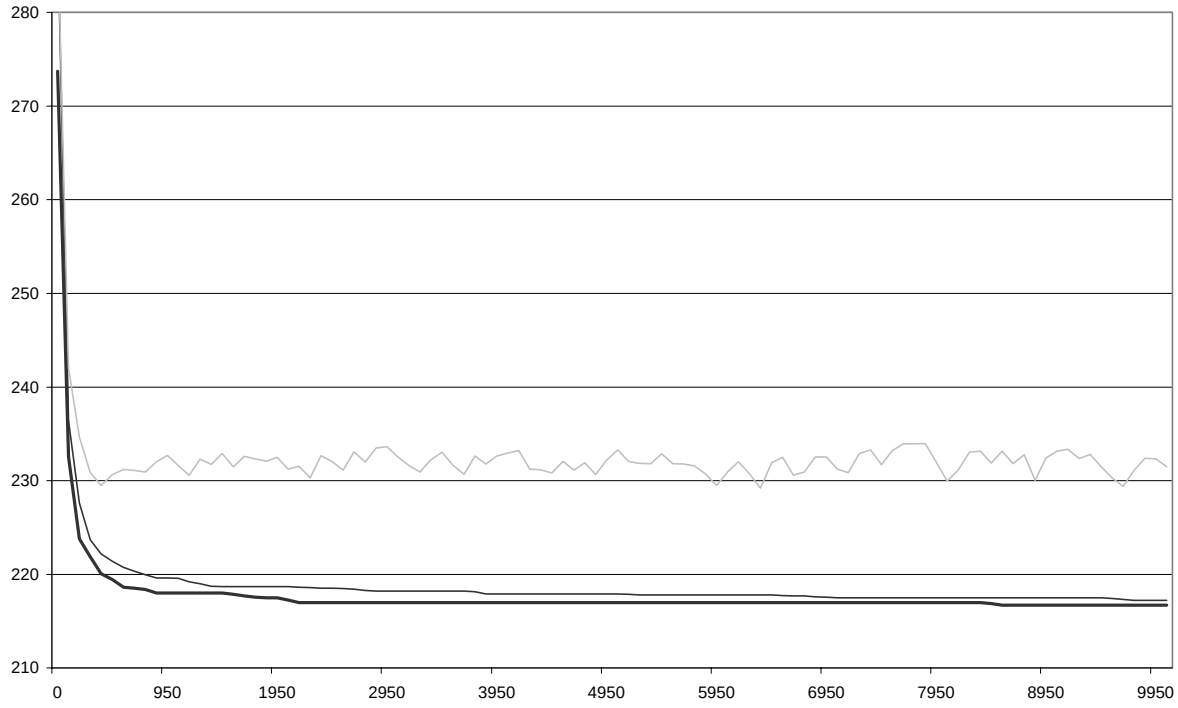

**Graf 1:** $\left|\overline{\Sigma}\right| = 2$



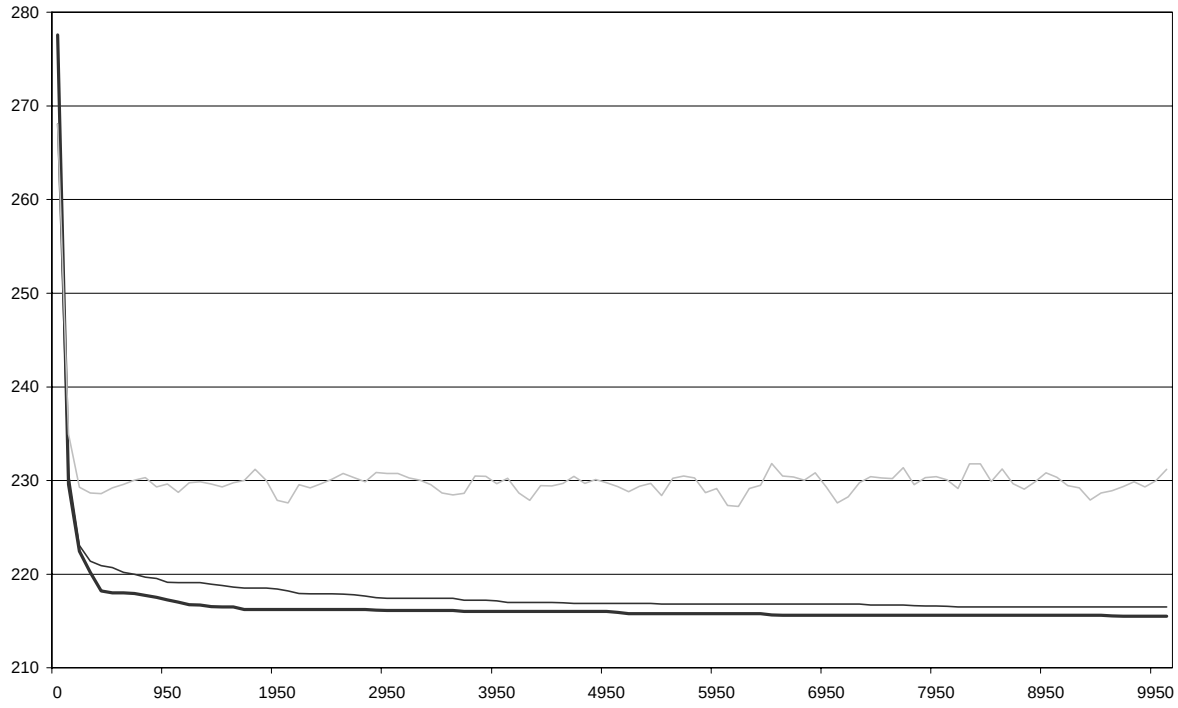

**Graf 2:** $\left|\overline{\Sigma}\right| = 5$



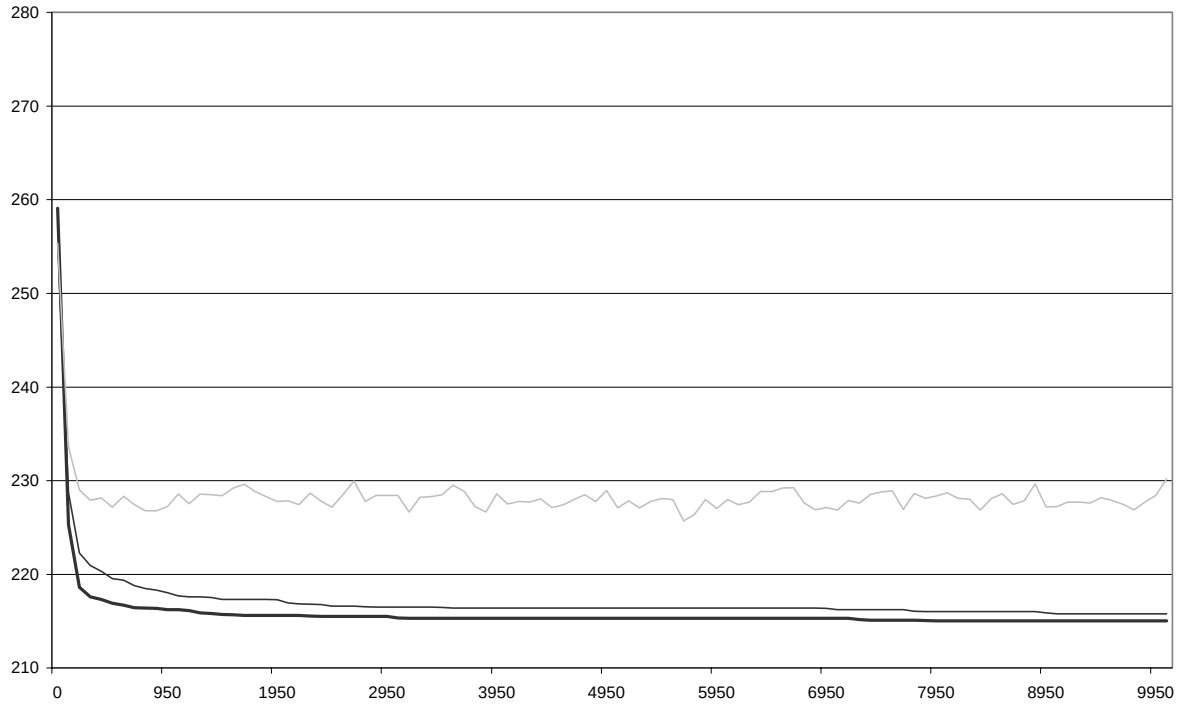

**Graf 3:** $\left|\overline{\Sigma}\right| = 10$



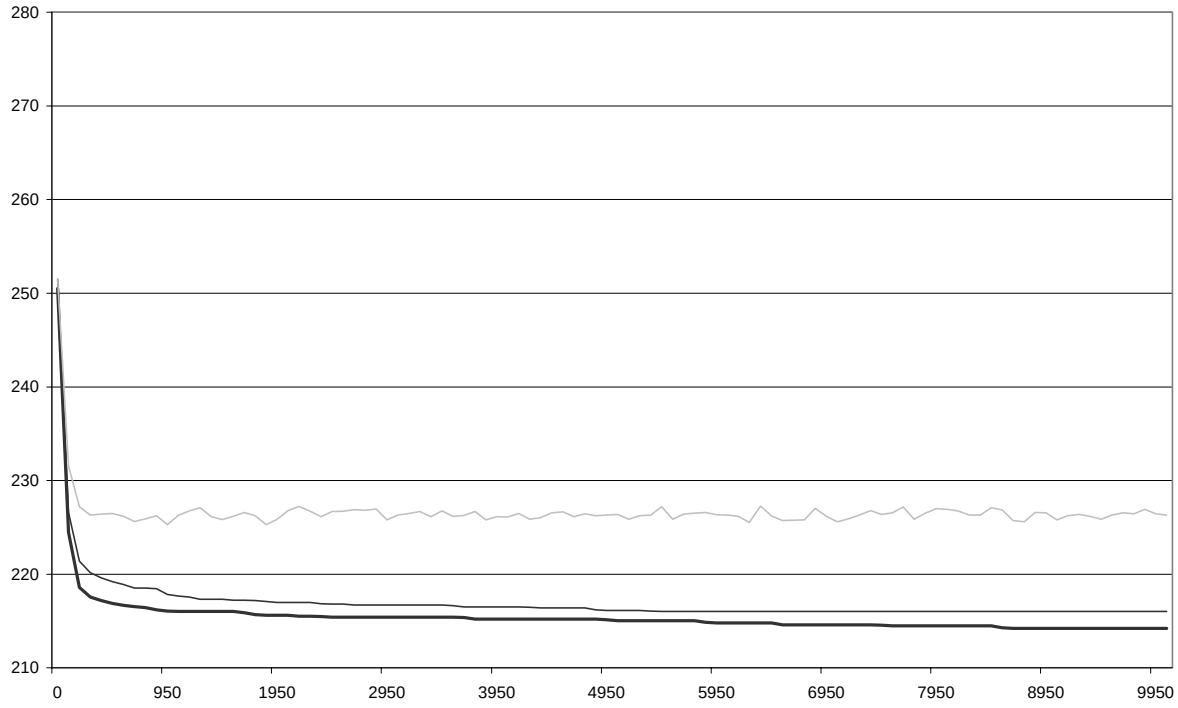

**Graf 4:** $\left|\overline{\Sigma}\right| = 25$



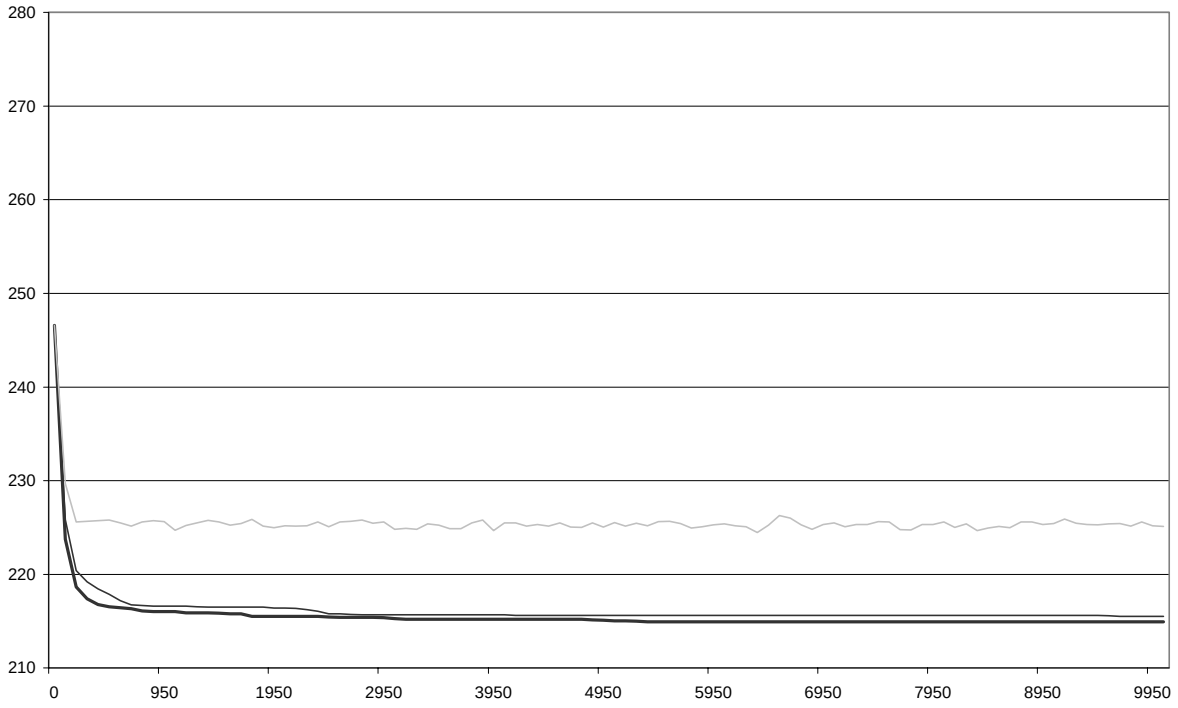

**Graf 5:** $\left|\overline{\Sigma}\right| = 50$




**Graf 6:** $\left|\overline{\Sigma}\right| = 100$




**Graf 7:** $\left|\overline{\Sigma}\right| = 250$




**Graf 8:** $\left|\overline{\Sigma}\right| = 500$