5. Řešení problému vážené splnitelnosti booleovské formule pokročilou iterativní metodou

Ladislav Martínek

1 Problém

Je dána booleovská formule F proměnnných $X = (x_1, x_2, \ldots, x_n)$ v konjunktivní normální formě (tj. součin součtů). Dále jsou dány celočíselné kladné váhy $W = (w_1, w_2, \ldots, w_n)$. Najděte ohodnocení $Y = (y_1, y_2, \ldots, y_n)$ proměnných $x_1, x_2, \ldots, x_n$ tak, aby $F(Y) = 1$ a součet vah proměnných, které jsou ohodnoceny jedničkou, byl maximální.

Je přípustné se omezit na formule, v nichž má každá klauzule právě 3 literály (problém 3 SAT). Takto omezený problém je stejně těžký, ale možná se lépe programuje a lépe se posuzuje obtížnost instance (viz Selmanova prezentace v odkazech).

2 Zadání úlohy

Problém řešte některou z pokročilých lokálních heuristik (simulované ochlazování, genetické algoritmy, tabu prohledávání). Řešení jinými metodami prosím zkonzultovat se cvičícím nebo přednášejícím. Volby konkrétních parametrů heuristiky a jejích detailů (operace nad stavovým prostorem, kritérium ukončení, atd. atd.) proveďte sami, tyto volby pokud možno zdůvodněte a ověřte experimentálním vyhodnocením. Hodnocení Řešení této úlohy je podstatnou součástí hodnocení zkoušky (28 bodů ze 100). Hodnotí se především postup při aplikaci heuristiky, tj. postup a experimenty, jakým jste dospěli k výsledné podobě (parametry, konkrétní operátory apod.). Například, pokud máte v řešení nějaké hodně neortodoxní prvky a pokud máte jejich výhodnost experimentálně doloženou, těžko mohou vzniknout námitky. Méně významné jsou konkrétní dosažené výsledky. Nežádáme rozhodně, aby semestrální práce měla úroveň světové výzvy Centra diskrétní matematiky Rutgersovy univerzity.

Tato práce by měla sloužit jako ověření Vašich schopností používat zvolenou pokročilou iterativní metodu. Ideálním výstupem by měl být algoritmus schopný řešit co nejširší spektrum instancí s rozumnou chybou. To neznamená, že pokud se Vám některé instance "nepovedou", je vše špatně. Důležité je, abychom viděli, že jste se aspoň snažili. Někdy to prostě nejde...

3 Rozbor zadaného problému

Úkolem semestrální práce je vytvořit program řešící problém vážené splnitelnosti boolovské formule v konjunktivní normální formě. Tedy řešit SAT problém s váhami jednotlivých proměných.

Hlavním kritériem pro mě bude splnění takového problému, tedy splnění všech klauzulí. Toto se ne vždy může podařit proto se případně budu snažit zachytit procento splněných klauzulí. Dalším kritériem bude váha dané instance podle ohodnocení proměných, která by měla být co nejvyšší.

Problém budu řešit pomocí genetického algoritmu, který popíši v další kapitole. Aby algoritmus mohl s řešeními pracovat je nutné vytvořit fitness funkci, pomocí které bude možné jednotlivé řešení porovnávat a rozhodovat o tom, které je úspěšnější. Tuto funkci jsem nejprve vytvořil jako součet poměru počtu splněných klauzulí a váhy ku maximální možné váze. K tomuto jsem byl následně nucet přidat koeficient ovlivňující přínos každého z kritérií, protože jak jsem psal výše splnění

problému je primárním cílem. Vytvořená fitness funckce vypadá následovně:

$$fitness(S) = a * \frac{\text{počet splněných klauzulí}}{\text{celkový počet klauzulí}} + (1 - a) * \frac{\text{váha řešení}}{\text{maximání možná váha}}$$

kde S je jedno řešení a a koeficient určující poměr členů, kde vysoká hodnota a blízká k 1 bude upřednostňovat řešení s větším počtem splněných klauzulí.

V rámci řešení budu pracovat i s konfiguracemi, které nejsou řešením, protože prostor řešení je obrovský a jen pár konfigurací splňuje kritérium úspěšnosti řešení a při jeho oříznutí bych mohl dostat nespojitý prostor, kde bych mohl často uváznout v lokálním minimu.

4 Popis implementovaného genetického algoritmu

Implementovaný genetický algoritmus je algoritmus, který se skládá z hlavního generačního cyklu a příslušných method, mezi hlavní patří selekce, mutace, křížení. Algoritmus je možné rozšířit i o mnoho dalších method např. elitismus nebo obměnu populace. Algoritmus je iterativní, kde počet cyklů může určovat předem daný počet nebo může být zastaven nějakou heuristikou. V práci jsem použil pevný počet cyklů, který je možné zadat. V každém cyklu je spočítána hodnota fitness funkce pro každého jedince. Dále jsou pomocí selekce vybíráni jedinci do nové populace. Tito jedinci mohou být kříženi a další vliv na ně má i mutace. Pomocí těchto method je možné udržet divezifikovanou populaci, pomocí které je probledáván stavový prostor. Popis jednotlivých metod:

Elitismus Je výběr nejlepších jedinců, kteří jdou do nové populace. Tento princip může vést k rychlé konvergenci populace.

Selekce Selekci jsem implementoval pomocí turnaje. Selekční tlak je řízen velikostí turnaje, kde malé turnaje davájí šanci vybrat i slabé jednice. Tedy toto je výhodně spíše na začátku, proto jsem v algoritmu vytvořil možsnot nastavení postupného růstu velikosti turnaje s každou generací. Jedinci sou do turnaje vybíráni náhodně s opakování přes celou generaci.

Křížení Dva jedinci vybraní selekcí můžu být s nějakou pravděpodobností kříženi. Z křížení vzniknou nový potomci, kteří budou tvořit základ nové generace. Křížení je možné provádět více způsoby, kde jsem zvolil dvoubodové křížení.

Mutace Je určena pravděpodobností změny každého bitu jedince. Provádí se pro každého jedince v nové generaci.

5 Implementace

Algoritmus jsem implementoval v jazyce Python a pro hlavní cyklus genetického algoritmu jsem využil rožšiření Cython, kde je psát kód Pythonu typovaný a je překládán do jazyka C pro spojení efektivity psaní v jazyce Python a rychlosti jazyka C.

Řešič příjímá v argumentech configurační soubor ve formátu yaml. V tomto souboru je podle vzoru (přiložen jako config.yml) nutné specifikovat složku s problémy, které bude algoritmus řešit a složku, kam bude ukládán výstup (csv soubor a grafy pro jednotlivé instance) a dále je také parametry samotného genetického algoritmu, které je možné zadávat jako počátek, konec a krok nebo jako jedinou hodnotu, pro rozsah budou vyzkušeny všechny zadané hodnoty.

Jednotlivé generace ukládám jako 2D pole, takto ukládám i novou generaci do které jsou přídaní buď vybraní jedinci nebo v případě křížení jejich potomci. Klauzule a váhy jsou také uloženy pomocí polí. Pro každou generaci tdy iteruji tak dlouho doku nenaplním potřebný počet jedinců, který byli určeny v konfiguraci.

Turnaj je prováděn vybíráním počtu náhodných indexů, kde je z jedninců ponechán pouze nejlepší, který je vrácen. Přes všechny generace je také ponecháno nejlepší řešení, které ovšem nezasahuje do jednotlivých generací. Dále jsou také ukládány jednotlivé hodnoty fitness pro každého

jedince v každé generaci, tak aby bylo možné vykreslit grafy průběhu algortimu. Parametry, které je možné nastavit pro genetický algoritmus jsou nasledující:

generationsize a

generationcount a

mutation a

crossover a

selection a

selection_add a

elitism a

fitness a

6 Testovací data

7 Testovací platforma

7.1 Cíle

Vlastnosti algoritmu lze měnit pomocí parametrů, jako je počáteční teplota, ochlazování a počet iterací na jedné teplotě. Při řešení tohoto problému, jsem neuvažoval neplatná řešení, jelikož začínám s prázdným batohem. Cílem je otestovat chování algoritmu v závislosti na těchto parametrech a vyvodit závěry. Tedy se naučit iterativní heuristiku nastavit na tento problém a řešit ho pomocí ní.

8 Experimenty

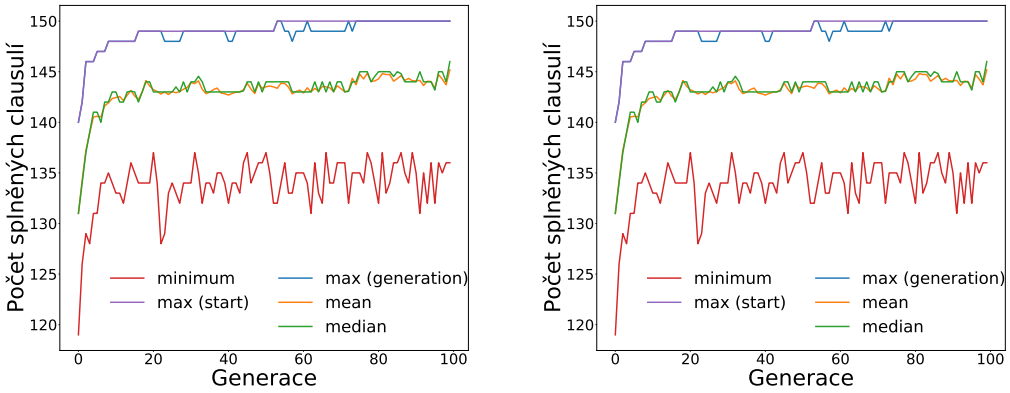
Experimenty jsem prováděl v režimu jednoho vlákna na starším datovém serveru v podobě starého notebooku, který v době výpočtu nebyl používán. Výsledky tedy nejsou ovlivněny jinými běžícími programy. Procesor na testovacím stroji: *Intel Pentium T3400 (2 cores). Taktován na 2.16 GHz s 1 MB cache.* Měření času CPU probíhalo v knihovně *timeit*. Algoritmus jsem napsal v Cythonu a časy tedy nemohou být případně srovnávány s řešením v předchozích úlohách.

9 Závěr

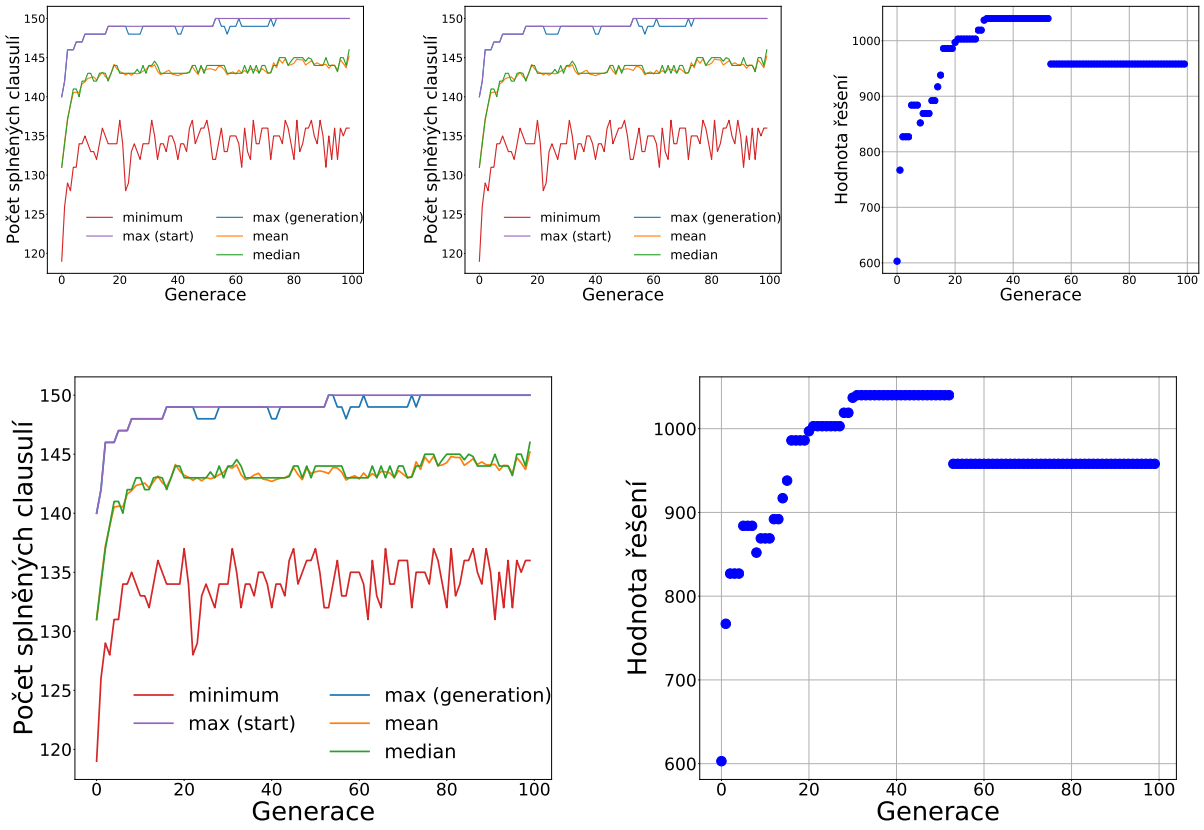
Během experimentu jsem prozkoumal pokročilou iterativní heuristiku - simulované ochlazování. Ověřil a prozkoumal jsem závislosti této heuristiky na řídících parametrech. Parametry jsou určitě závislé na daných problémech či paramentech instancí, což je patrné i ze vzorců, kde přímo vystupuje rozdíl cen řešení. Tyto rozdíly se mohou pohybovat v různých intervalech a tomu je potřeba parametry také upravit, tedy hýbat s počáteční teplotou, která ve vzorci vystupuje jako druhý parametr.

Simulované ochlazování je randomizovaná heuristika sloužící k prochazení prostoru, a proto může při špatném nastavení mít tendenci uváznutí v lokálních extrémech. Heuristika kombinuje přístup diverzifikace na záčatku, a následující intenzifikace je snaha o nalezení optimálního řešení.

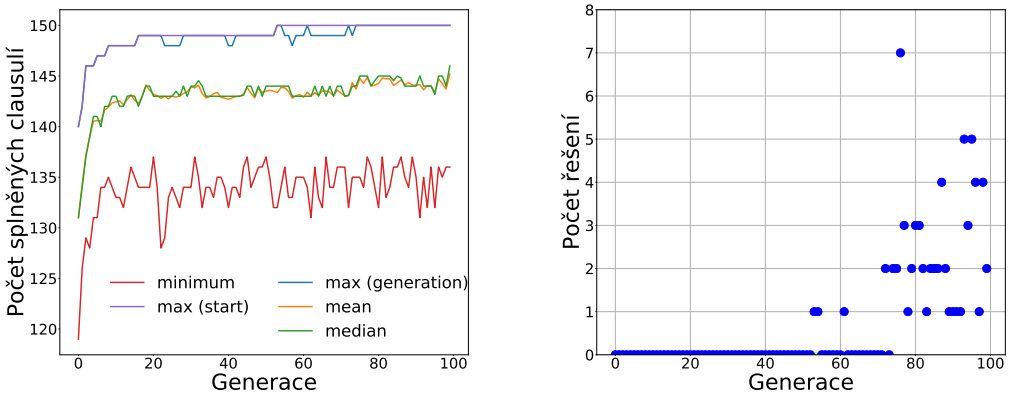
Ze závislostí zjištěných během experimentu je vidět, že počáteční teplota je důležitý parametr v závislosti na hodnotách ceny řešení. Pokud se ceny řešení pohybují ve velkých hodnotách bude mou snahou nastavit vyšší teploty, než pokud se budou pohybovat na nějakém intervalu a budou například normalizované.



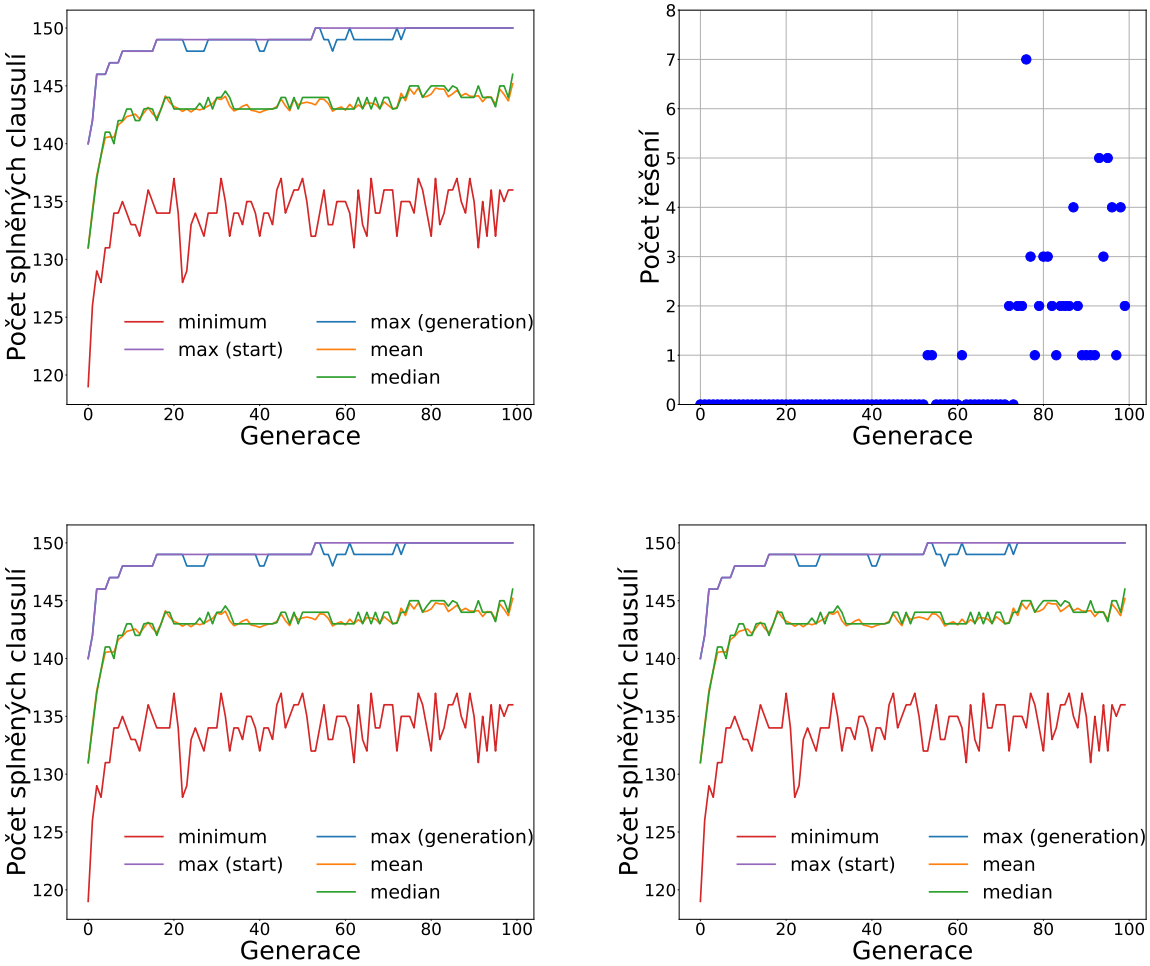
Obrázek 1: Na levém grafu je závislost relativní chyby na počáteční teplotě. Na pravém grafu je závislost výpočetního času na počáteční teplotě



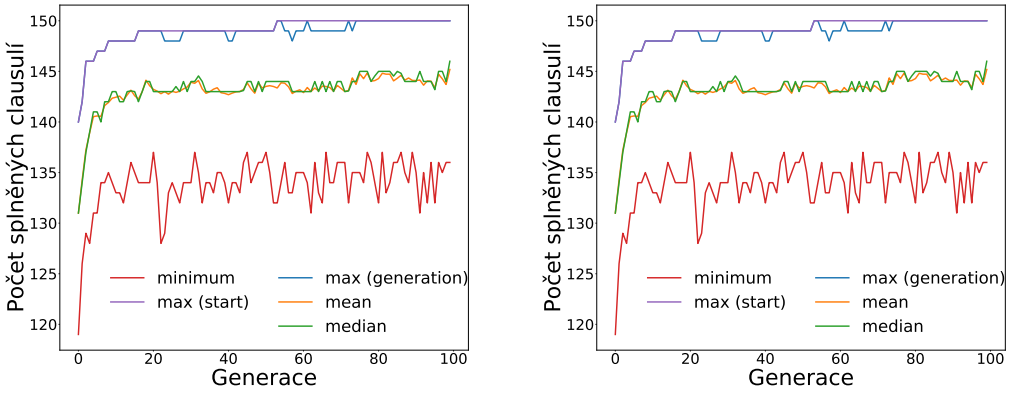
Obrázek 2: Zde jsou uvedené grafy vývoje řešení pro vybrané hodnoty počtu iterací na jedné teplotě. Konkrétně zleva pro hodnoty 30, 60, 120, 240, 300



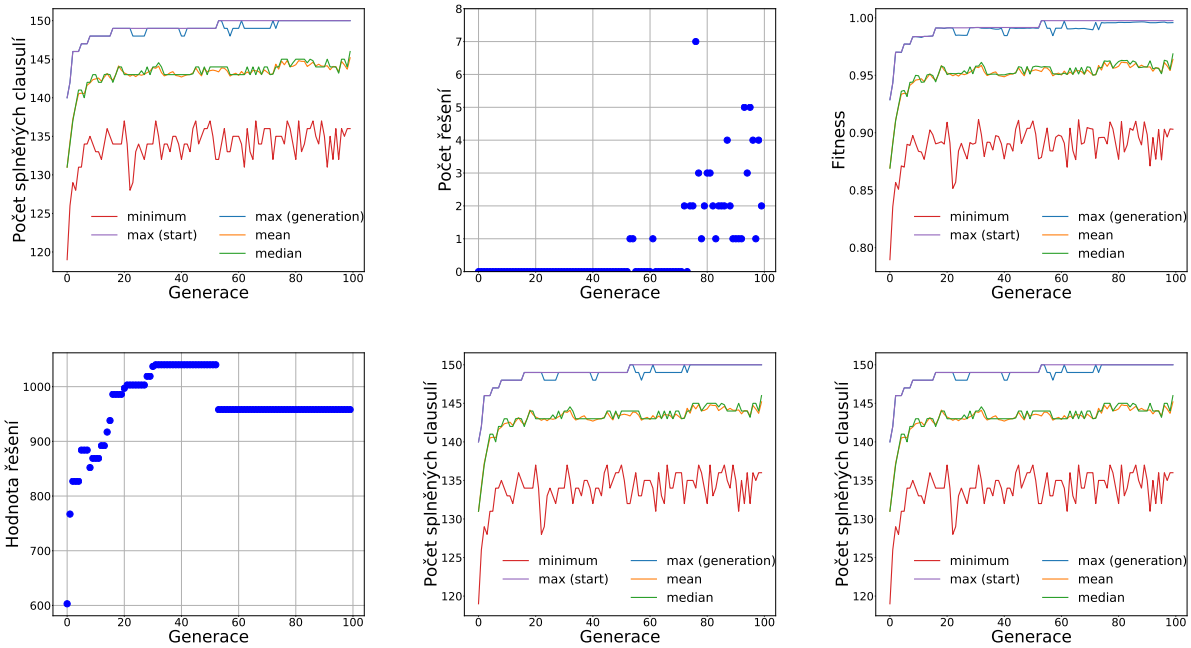
Obrázek 3: Na levém grafu je závislost relativní chyby na koeficientu ochlazování. Na pravém grafu je závislost výpočetního času na koeficientu ochlazování



Obrázek 4: Zde jsou uvedené grafy vývoje řešení pro vybrané hodnoty koeficientu ochlazovaní. Konkrétně zleva pro hodnoty 0.993, 0.995, 0.997, 0.999



Obrázek 5: Na levém grafu je závislost relativní chyby na počtu iterací na jedné teplotě. Na pravém grafu je závislost výpočetního času na počtu iterací na jedné teplotě



Obrázek 6: Zde jsou uvedené grafy vývoje řešení pro vybrané hodnoty počtu iterací na jedné teplotě. Konkrétně zleva pro hodnoty 30, 60, 120, 180, 240, 300

Parametr ochlazování je důležitý k dostatečému na vzorkování rozsahu teploty a tedy i dostatečném počtu kroků v jednotlivých fázích a to především ve fázi intenzifikace.

Jednotlivé počty kroků na dané teplotě nám naopak dovolí prozkoumat dané okolí, ale tento parametr nemá lineární závislost na chybě, ale je dán nepřímou uměrou a volíme tak mezi relativní chybou a časovou náročností.

Časová náročnost algortitmu je daná nastavenými parametry a s konkrétními parametry provede algoritmus vždy stejný počet kroků, tedy pro mou implementaci tohoto přístupu. Simulované ochlazování je možné například implementovat s proměným počtem kroků na jedné teplotě, který může být opět řízen nějakou jednoduchou heuristikou.