Univerzita Pardubice

Fakulta ekonomicko-správní

Metoda Tabu search ve zvoleném softwarovém prostředí

Martin Svoboda

Bakalářská práce

2009

Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Martin SVOBODA, DiS.**

Studijní program: **B6209 Systémové inženýrství a informatika**

Studijní obor: **Informační a bezpečnostní systémy**

Název tématu: **Metoda Tabu search ve zvoleném softwarovém prostředí**

Zásady pro vypracování:

1. Charakteristika metody Tabu search
2. Algoritmus a kód v prostředí Visual Basic
3. Modely testování metody Tabu search

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Glover, F. Laguna, M. Tabu Search, Boston : Kluwer Academic Publishers, c1997. ISBN 0-7923-8187-7

Glover, F. Tabu Search — Part I, ORSA Journal on Computing, c1989

Glover, F. Tabu Search — Part II, ORSA Journal on Computing, c1990

Cvijovic, D. Klinowski, J. Taboo search - an approach to the multiple minima problem, Science, c1995

Vedoucí bakalářské práce: **Ing. Jan Pavuš, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **6. října 2008**

Termín odevzdání bakalářské práce: **1. května 2009**

doc. Ing. Renata Vykoupil, Ph.D.
děkanka

L.S.

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 6. října 2008

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 27. dubna 2009

Martin Svoboda

Na tomto místě bych rád poděkoval Ing. Janu Panušovi, Ph.D., vedoucímu mé bakalářské práce, za poskytnutí cenných rad při tvorbě tohoto textu a čas, po který se mi věnoval v rámci konzultací.

Anotace

Tato práce se zaměřuje na problematiku optimalizace využitím metody zakázaného prohledávání. V úvodních částech popisuje teoretické základy této metody a jejich klíčových elementů. V dalších částech se zabývá implementací v prostředí Microsoft Access, kde je výsledkem vytvořený algoritmus společně s daty získaných v průběhu testování.

Klíčová slova

zakázané prohledávání, optimalizační algoritmy, metaheuristika, globální optimum, krátkodobá paměť

Title

Tabu search method

Anotation

This work is aimed at the problems of optimalization by using Tabu search method. In the introduction are described the theoretic principles of this method and their key elements. Next chapters deal with the implementation in Microsoft Access, the result of it is the algorithm created by testing together with the dates obtained during this process.

Keywords

Tabu search, optimization algorithms, metaheuristics, global optimum, short-term memory

Obsah

1 Úvod

S problémy optimalizace se setkáváme prakticky v každém oboru lidské činnosti. Každý z nás často zvažuje, jakým nejlepším způsobem vyřešit daný problém. Pro různé problémy existují různé postupy řešení. Jedním z nich je metoda Tabu search, kterou se budeme v této práci zabývat. Algoritmus Tabu search, neboli zakázané prohledávání, nachází uplatnění zejména při řešení problémů, na které neexistuje exaktní algoritmus, anebo které jsou pro daný algoritmus příliš rozsáhlé.

Cílem práce je implementace této metody v prostředí Microsoft Access a seznámení čtenáře s problematikou optimalizace TS.

Tato práce popisuje techniku zakázaného prohledávání v klasickém spojení s horolezeckým algoritmem. V praxi je tato metoda úspěšně používána na řešení problému obchodního cestujícího, návrhu komunikačních sítí, nebo logistických problémů.

První část objasňuje techniku Tabu search na řešení problému obchodního cestujícího a zavádí pojem tzv. krátkodobé paměti, která je pro tuto metodu charakteristickým znakem. Jelikož se jedná o velmi rozsáhlou problematiku, jsou zde popsané jen základní prvky zakázaného prohledávání, jmenovitě typy paměti, velikost zakázaného seznamu a formy prozkoumávání okolí.

Důraz je kladen zejména na ukázání souvislostí s vyhledáváním lokálního minima, kterému je věnována další část. Zde je započata implementace algoritmu a nastíněna souvislost s prvky zkoumané metody a jejími proměnnými. V další fázi je algoritmus doplněn o paměť a následuje testování na jednorozměrné funkci. Právě daný rozměr funkce poukazuje na formu prozkoumávání určitého okolí. Z výpočtů jednorozměrné funkce vychází poslední kapitola, testování na dvourozměrné funkci, jež je zaměřena na určení velikosti zakázaného seznamu a způsobu prohledávání prostoru.

Výsledkem části praktické je demonstrace funkčnosti algoritmu při optimalizaci Rastriginovy a Schwefelovy funkce využívané pro testování optimalizačních algoritmů.

2 Úvod do metody

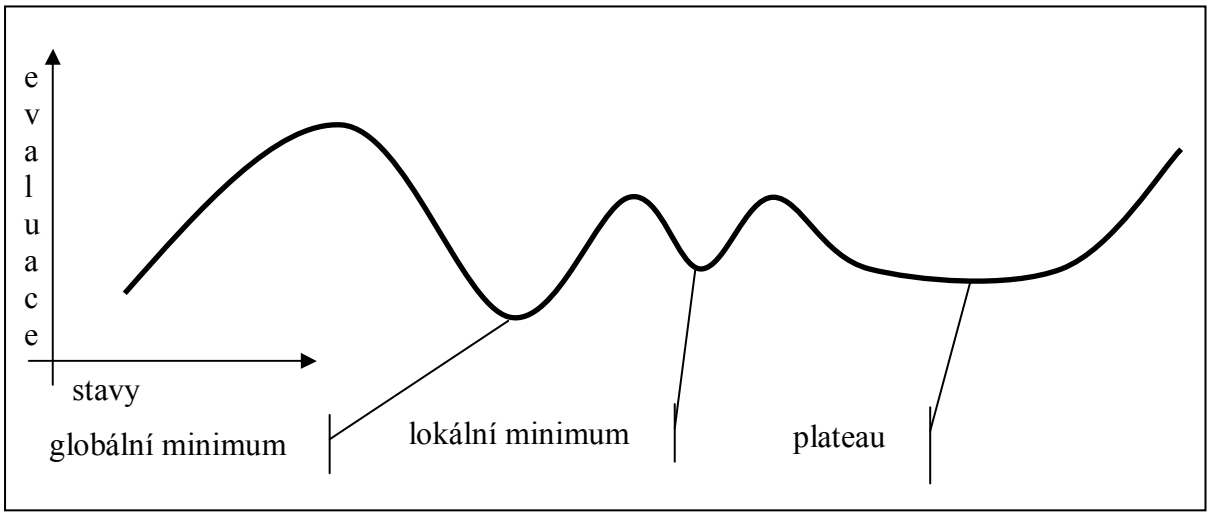
Tabu search je meta-heuristika (deterministická optimalizační metoda), která řídí lokální heuristickou vyhledávací proceduru k prozkoumání řešeného prostoru přes lokální optimum [1]. Lokální proceduru představuje vyhledávání, které využívá operace nazývané tah k určení okolí z daných řešení. Jedna ze součástí Tabu prohledávání je použití jeho adaptivní paměti, která vytváří flexibilnější režim vyhledávání. Strategie založené na paměti jsou tedy charakteristickým znakem přístupu zakázaného prohledávání [2].

Tabu search, ve zkratce TS, se dá považovat za „nadstavbu“ heuristických procedur pro řešení optimalizačních úloh, navržený k řízení metod (nebo jejich zpracovávaných součástí) sloužících k úniku z pastí lokálního minima.

Společně se simulovaným žíháním a generickými algoritmy bylo TS v širším pojetí oceněno komisí CONDOR v roce 1988 jako „mimořádně nadějný“ pro budoucí použití v praxi. Následující prudký a trvalý růst použití zakázaného prohledávání dostatečně potvrdil toto ocenění. Čisté a smíšené přístupy stanovily nové výsledky v hledání lepšího řešení v oblasti výrobního plánu a plánování, přidělování zdrojů, návrhu sítí a směrování v telekomunikacích mezi mnoha dalšími oblastmi.

3 Technika Tabu search

Tabu search se dá stěží považovat za techniku nadpřirozenou, zabývá se efektními omezeními k usměrňování vyhledávacích procesů k překonání jinak obtížných oblastí. Tato omezení se dají nazvat jako „zakázaná“ a jsou vytvářena nebo ukládána do pamětí k tomu určených.



Obr. 1 Terminologie [3]

Metoda TS je založena na předpokladu, že problém je řešitelný. Aby se dala označit za inteligentní, musí být začleněna adaptivní paměť a responzivní hledání. Vhodná analogie této metody by se dala použít při horolezeckém výstupu, kde si lezec musí selektivně pamatovat klíčové prvky své cesty (užití adaptivní paměti) a musí být schopen provést strategické volby v průběhu své cesty (užití responzivního hledání). Na obrázku 1 je pro představu zobrazen průběh funkce s jednotlivými stavy, které mohou nastat. Hlavní roli pro adaptivní paměť (jejíž význam je navrhnout analogicky k horolezci, který musí vyhodnotit aktuální možnosti ve vztahu k následujícímu stoupání podobného terénu) poskytuje realizaci postupů, které jsou schopny vyhledávat možná řešení ekonomicky a efektivně. Jelikož lokální výběry jsou řízeny informacemi sbíranými v průběhu vyhledávání, TS je v rozporu s bezpaměťovými typy, které jsou založeny na náhodném výběru. Adaptivní paměť je tedy rozdílná oproti pevné paměti, která je typická pro větvení a hraniční taktiky.[4]

Velký důraz je kladen na responzivitu, v češtině by se dal použít ekvivalent citlivost hledání (a tudíž účel) v zakázaném prohledávání, který v deterministických a pravděpodobnostních provedeních čerpá domněnky, že „špatné strategické rozhodnutí může přinést více informací než dobrá náhodná volba“ [2]. V systému kde je použita paměť, může špatné rozhodnutí založené na strategii poskytnout užitečné informace o tom, jak lze strategii výhodně změnit.

4 Kategorie pamětí v TS

Struktura paměti v zakázaném prohledávání se dá dle [2] rozdělit do čtyř kategorií, a to pamětí založených na:

aktuálnosti (Recency-based),

četnosti (Frequency-based),

kvalitě (Quality),

vlivu (Influence).

Aktuální, neboli krátkodobá paměť, zaznamenává čas nebo počet iterací použitých v průběhu prohledávání. Tomuto typu paměti je v této práci věnována samostatná kapitola. Paměť četnosti (tzv. dlouhodobá) zaznamenává, jak často byl prvek během prohledávání použit.

4.1 Kategorie kvality

Kategorie kvality se vztahuje ke schopnosti rozlišovat hodnoty řešení navštívených během hledání. Tento charakter paměti tedy určuje, zda jsou jednotlivé prvky dobrým řešením, anebo zda alespoň k takovýmto řešením vedou. Provozně pak slouží jako základ k učení, kde tyto podněty slouží k tzv. pokutování resp. odměňování, tj. k zesílení akcí, které vedou k dobrým řešením a trestům sloužícím k zabránění akcí, jež vedou k řešením špatným. Přizpůsobivost této paměti dovoluje hledání ve více-objektovém prostředí, kde výsledek směru hledání může být definován více než jednou funkcí.[1]

4.2 Paměť vlivu

Takzvaná paměť vlivu *(*Influence memory*)* se v TS zabývá strukturou - složením výběru získaného během prohledávání. Zaznamenávání informací o vlivu vybraných prvků přidává další úroveň učení. Tento typ paměti slouží k určení pravidel pro oddělování uzlů nebo pro zachování směru větvení.

4.3 Typy pamětí

Paměť používaná v zakázaném prohledávání je explicitní a atributivní. Explicitní paměť zaznamenává soubor řešení, typicky se skládajících z vybraných řešení získaných během hledání. Zvětšení této paměti vede k zaznamenávání velmi atraktivních prvků v okolí, které jsou pak použity k rozšíření lokálního prohledávání. Do atributivní paměti se zaznamenávají informace o atributech, které vedly k posunu z jednoho řešení do dalšího.

5 Intenzifikace a diverzifikace

Dvě velmi významné součásti zakázaného prohledávání jsou taktiky intenzifikace a diverzifikace. Dá se říci, že obě tyto metody jsou určitou formou prozkoumávání. Intenzifikace je založena na přizpůsobení kandidátních pravidel k úpravě kombinací a řešení tahů. Může tedy způsobit návrat do oblasti proto, aby došlo k důkladnějšímu prohledávání [2]. Jako příklad by se dal použít zakázaný tah, který vede k dosud nejlepšímu řešení. Vzhledem k tomu, že vybraná řešení jsou zaznamenána na zkoumání svých přímých sousedů, explicitní paměť má přímou souvislost s realizací intenzifikace. Fáze intenzifikace se zaměřuje na zkoumání okolí nejlepšího řešení, jinými slovy by se dala nazvat jako konvergence k finálnímu řešení. Typickým příkladem použití intenzifikace je znovuspuštění algoritmu v bodě, kde předchozí iterace docílila nejlepšího řešení [4].

Termín sousedů má zde širší význam než jen obvyklé prohledávání okolí. Kromě předpokládaných řešení, která jsou sousedy nebo blízko elitních řešení (získanými běžnými postupy), proces intenzifikace zahrnuje také dobrá řešení, anebo takové prvky, které vznikly použitím upravených ohodnocení.

Naproti tomu fáze diverzifikace slouží v procesu vyhledávání k prozkoumání nenavštívených oblastí a k rovnoměrnému průzkumu stavového prostoru [3]. Kompletní řešení může být v diverzifikaci vytvořeno z jednotlivých částí jeho součástí, nebo může být vytvořeno z upravených vyhodnocování, jako je například použití penalt nebo výhod.

5.1 Programování adaptivní paměti

Zakázané prohledávání se v posledních letech stalo zájmem mnoha srovnávacích studií a uplatňování v praxi. Následkem toho se začaly objevovat problémy spojené s řešením složitých optimalizací. Ne všechny tyto problémy byly použitím metody TS vyřešeny. Důvodem pro tento dvojsmysl je, že metoda byla v literatuře představena dvojím způsobem, tedy jako dvě rozdílné metody – první z nich jednodušší, druhá jako pokročilá. Jednodušší metoda obsahuje pouze omezenou část TS techniky. Je často používána v přípravných fázích k otestování omezené podmnožiny, a obvykle pracuje pouze s krátkodobou pamětí. Pokročilejší metoda zahrnuje dlouhodobou paměť společně s metodami intenzifikace a diverzifikace. Tento druhý přístup se díky jeho zaměření na využití a sbírání prvků z paměti někdy nazývá jako „programování přizpůsobivé paměti“. Tedy dlouhodobá paměť je důležitá k získáni nejlepších výsledků v obtížných úlohách [5].

6 Užití krátkodobé paměti

Metoda zakázaného prohledávání může být použita přímo k rozhodovacím problémům, které jsou vyjádřeny slovně nebo symbolicky, aniž by bylo nutné je transformovat do matematických formulací. Pro představu zde bude uvedeno fungování TS pomocí matematického zápisu. Skupinu úloh označíme jako optimalizující (minimalizující nebo maximalizující) funkci f(x), kde $x \in X$. Funkce f(x) může být lineární i nelineární, množina X představuje vektor vybraných proměnných x.[6]

TS začíná stejným způsobem jako běžné lokální hledání souseda, vychází iteračně z jednoho bodu k dalšímu, dokud nejsou splněny podmínky ukončení. Každé x e X má přidruženého souseda $N(X) \subset X$, a každého řešení $x' \in N(x)$ je dosaženo z x operací nazvaných tah.

Jako výchozí bod můžeme porovnat zakázané prohledávání se sestupnou (vzestupnou) metodou, kde je cílem minimalizovat (resp. maximalizovat) f(x). Tento způsob dovoluje pouze ty tahy, které vedou ke zlepšení hodnot funkce a skončí ve chvíli, kdy není žádná možnost dalšího zlepšení. Cílové x získané ze sestupné (vzestupné) funkce nazýváme lokálním optimem, neboť se jedná o nejlepší výsledek, anebo přinejmenším o lepší než všechna řešení v jeho okolí. Zřejmým nedostatkem sestupné metody je, že lokální optimum nebude ve většině případech optimem globálním, jak je obvyklé, proces minimalizace funkce x neproběhne přes všechny $x \subset X$. [7]

6.1 Pseudokód sestupné metody

Dle [5] je postup následující:

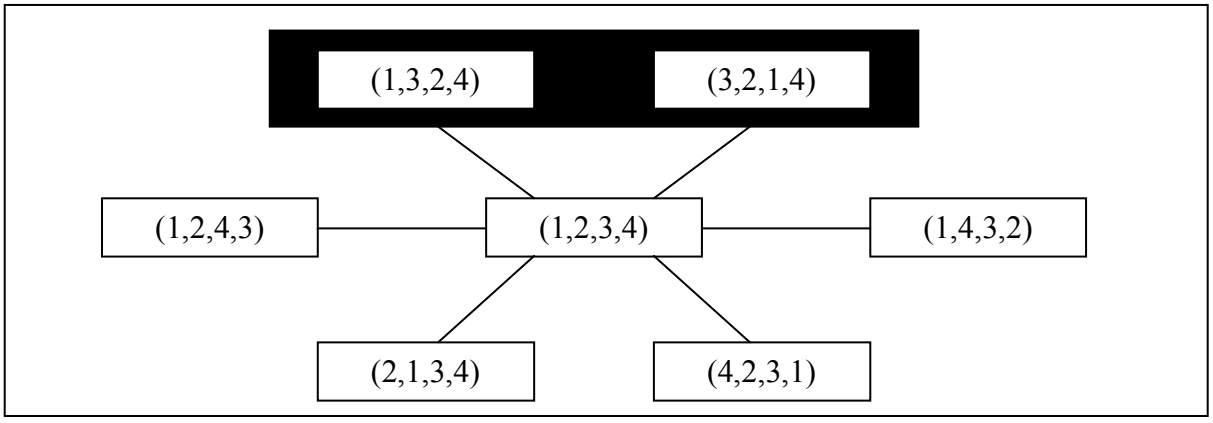
a) vyber $x \in X$ pro začátek zpracování
b) najdi takové $x' \in N(x)$, kde platí $f(x') < f(x)$
c) pokud neexistuje žádné x´, x je lokálním optimem a metoda je zastavena
d) jinak označ x´ jako nové x a jdi na bod b)

Pro výběr dobrého řešení z aktuálních „sousedů“ je významný mechanizmus řízení v TS, který slouží k tomu, aby se dostal přes bod lokálního optima. Čili první důležitý úkol pro TS je stanovit vhodný list kandidátů pro zúžení zkoumaných prvků z N(x) za účelem dosažení efektivního porovnání mezi kvalitou x´a úsilím vydaným k jeho nalezení.

6.2 Tabu klasifikace v krátkodobé paměti

Každá z obou typů pamětí (krátkodobá vs. dlouhodobá) má své vlastní speciální techniky. Může být viděno jako změny v „sousedství“ okolí N(x) aktuálního řešení x. Vybrané hodnoty, které byly získány během hledání, jsou výsledkem modifikovaného okolí, které značíme N*(x). V TS s krátkodobou pamětí N*(x) je podmnožinou N(x) a Tabu klasifikace slouží k identifikaci prvků N(x), které budou vyloučeny z N*(x). V dlouhodobém pak může být N*(x) rozšířen o hodnoty, které nebyly zpravidla nalezeny v N(x). Znamená to tedy, že okolí x není statická množina, ale že se může měnit podle historie hledání. U přístupu s krátkodobou pamětí může být řešení x prohledáno i vícekrát, ale je pravděpodobné, že příslušné zmenšené okolí bude vždy jiné. Z praktického hlediska tato metoda rozpozná nejvýhodnější, anebo blížící se nejvýhodnější řešení dlouho předtím, než bude prozkoumaná značná část z X[2].

Velmi důležité hledisko TS se týká výběru vhodného omezení N*(x). Kvůli využití paměti, N*(x) závisí na trajektorii vyplývající z posunu od jednoho řešení k dalšímu (nebo na skupině trajektorií v případě paralelního zpracování). Výchozí bod týkající se zkoumání krátkodobé paměti považujeme za tvar paměti začleněný v Tabu seznamu T který výlučně obsahuje t rozdílná řešení, což je $T=(x_1, x_2,....x_t)$, kde $N^*(x) = N(x)\backslash T$, t.j. N*(x) složených z řešení N(x), které nejsou obsaženy v tabu seznamu.[2]

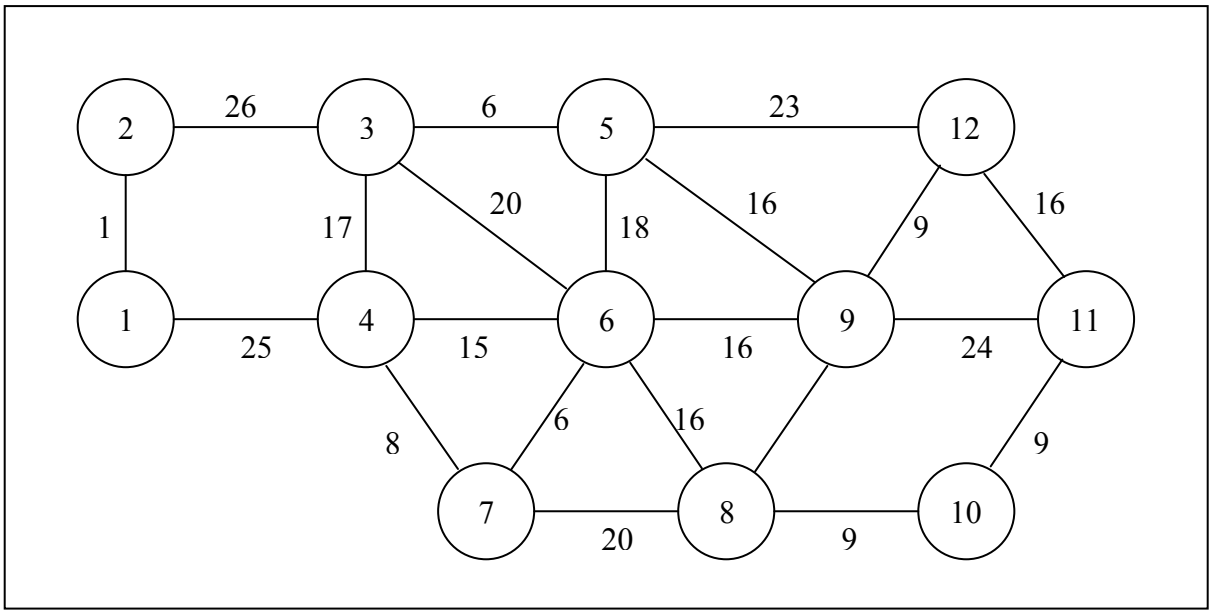


Obr. 2 Redukce okolí [2]

Jestliže aktuální řešení je dáno x=(1,2,3,4) a tabu seznam je složen z hodnot T={(1,3,2,4), (3,1,2,4), (3,2,1,4)}, potom modifikované okolí N*(x) obsahuje pouze 4 ze 6 řešení N(x). Obrázek 2 ukazuje, že jen dvě ze tří řešení v T patří do okolí x.

6.2.1 Aktuální paměť

Nejvíce používaná krátkodobá paměť sleduje atributy řešení, které se změnily při posledním průchodu a je nazývána aktuální pamětí**.** Vybrané atributy, které se vyskytují v navštívených řešeních, jsou označeny jako Tabu-aktivní, a řešení, která obsahují tabu-aktivní prvky, se stávají tabu**.** Tato funkce zabraňuje opětovnému prohledávání stejně tak jiným řešením, která mají podobné vlastnosti jako řešení označená tabu-aktivní, je zabráněno znovu procházet. Tabu hodnocení dokáže kromě poukazování na řešení která jsou zakázaná proto, že obsahují tabu-aktivní atributy, také odkazuje na tahy, které vedou k takovým řešením. Tato situaci je vysvětlena na příkladu minimalizace stromu. Tato metoda spočívá v prohledávání stromu na obrázku 3. složeného z cest v grafu tak, že suma vah těchto cest je minimum.



Obr. 3 Vážený neorientovaný graf [2]

Uzly jsou zde vyobrazeny jako číslované kruhy a cesty jako linky spojující pár uzlů. Váhy jsou čísla u cest..

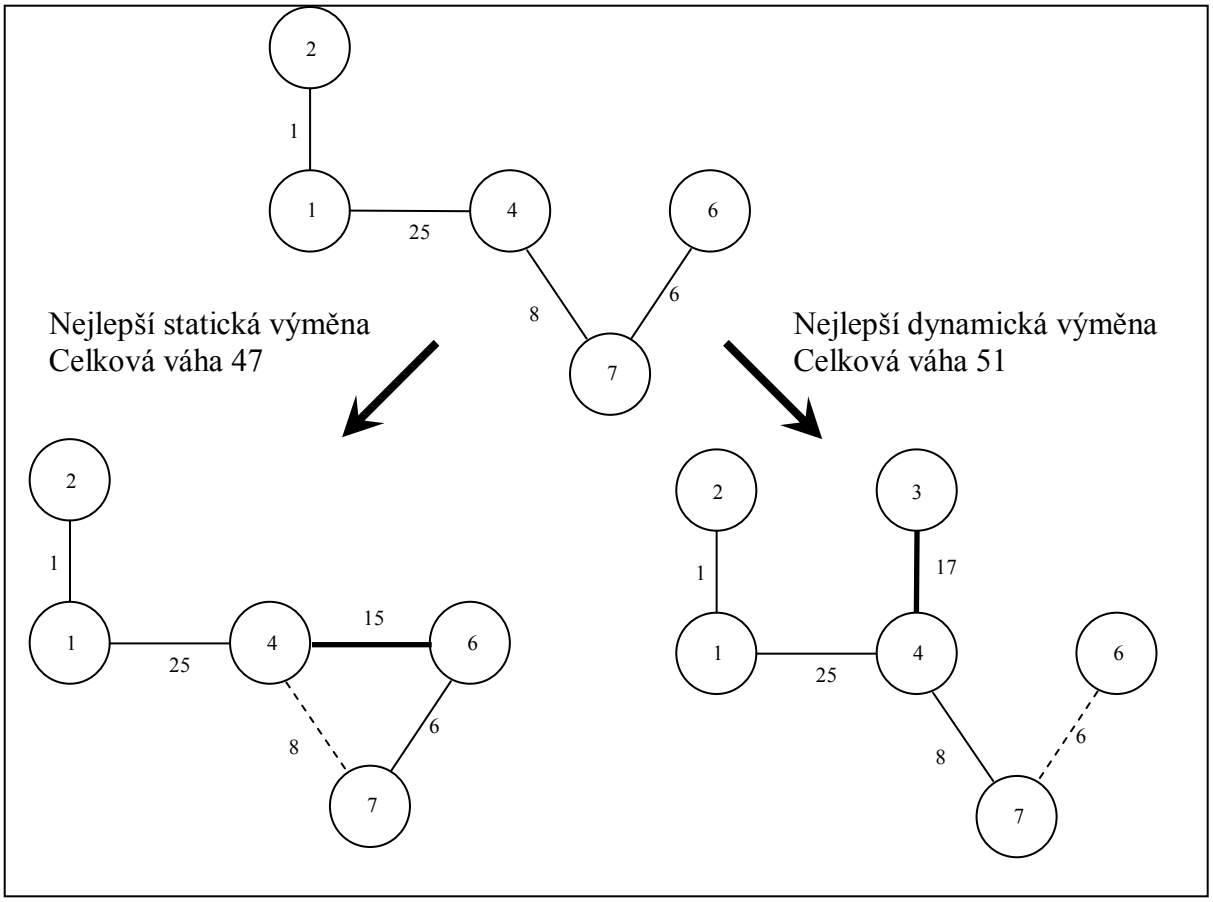
Konstrukce začíná výběrem cesty(okraje) (i,j) s nejmenší váhou v grafu, kde i a j jsou ukazatelé uzlů. Zbývající k-1 cesty jsou vybírány tak, aby zvyšování celkové váhy bylo v každém kroku minimalizováno. Vždy jsou vybírány navazující sousední uzly. Výsledek pro k=4 je popsán v tabulce 1.

Tab. 1 Prvky Hladové konstrukce [2]

Tabulka Hladové konstrukce			
krok	kandidáti	výběr	celková váha
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

Začátek – cesta mezi uzly (1,2) s vahou 1. Z kandidátských uzlů (2,3) a (1,4) vybrán ten s menší vahou, tzn. (1,4), iterativně pokračujeme, dokud není splněna podmínka stromu složeného ze 4 cest. Počáteční řešení v tomto případě je celková váha 40.

Mechanismus výměny tahu slouží k nahrazení vybraného okraje ve stromu jiným zvenku. Rozlišujeme dva druhy mechanismů – první z nich je takový, který nemění uzly, ale pouze cesty mezi nimi (statický), naproti tomu druhý má za následek nahrazení původního uzlu uzlem novým (dynamický).



Obr. 4 Typy výměny tahu [2]

Z obrázku 4 vychází lépe statická výměna tahu a má za následek zvýšení celkové váhy. Provedení tohoto pohybu ruší pravidla blížení ke klesání a nastavuje cyklus pro tabu zakázaný proces. Dalším krokem v TS je volba vhodných klíčových atributů, které budou použity pro tabu klasifikaci. V této problematice, kde jsou tahy definované přidáváním a mazáním prvků, může být označení těchto prvků použito jako atributů pro uplatnění tabu-stavu.

6.2.2 Výběr Tabu klasifikace

Tabu třídění nemusí být stejnoměrné. Je totiž navrženo tak, aby zpracovávalo přidané nebo zrušené prvky rozdílně. Tabu-aktivní status má různé významy v závislosti na tom, zda se jedná o přidávání nebo rušení spojení. Podobný rozdíl je nutno poznamenat v uchovávání tabu-aktivních rozhodnutích – zde je rozumné navrhnout tabu-sklad tak, aby udržoval po delší časový interval odebíraná spojení než ta přidávaná[8]. Velmi důležitá je pro tabu-aktivní fázi hodnota k, protože jestliže po tuto dobu, resp. počet kroků *k* nedojde ke klesání, potom všechny *k* kroky jsou klasifikovány jako tabu[2].

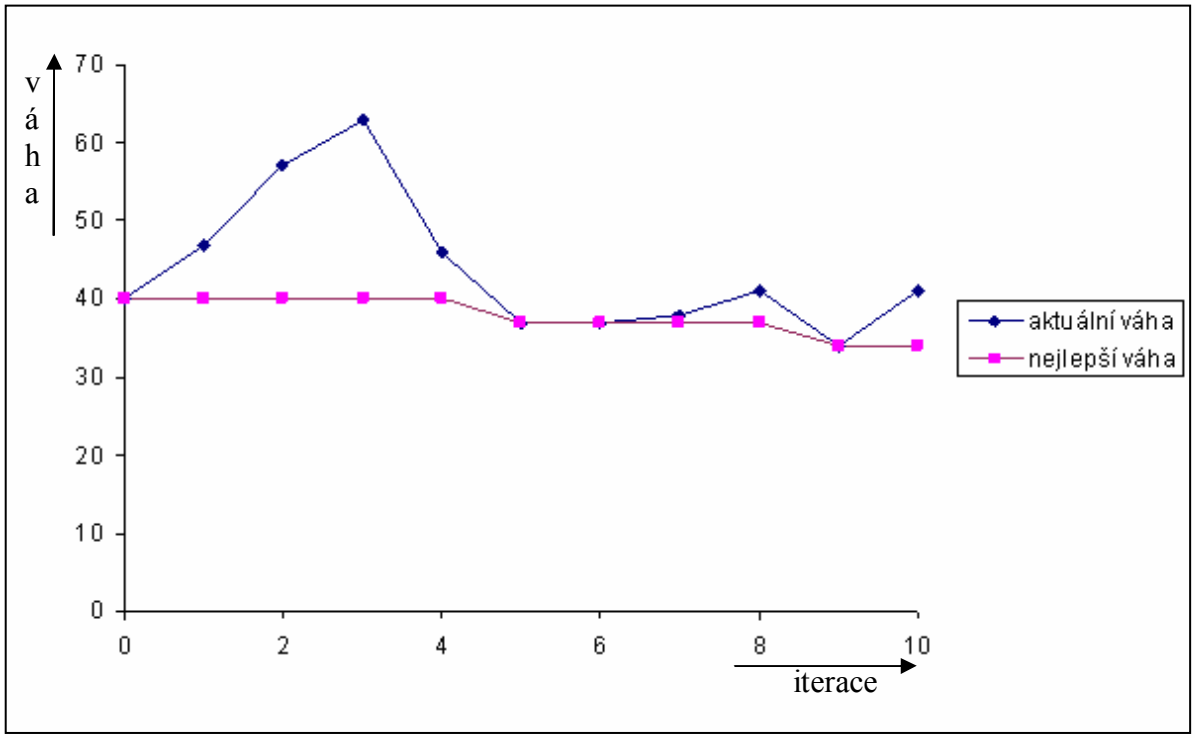
Tab. 2 TS iterace [2]

Tabu search iterace						
iterace	síť tabu držení		přidat	odebrat	hodnota pohybu	váha
	1	2				
1			(4,6)	(4,7)		47
2	(4,6)	(4,7)	(6,8)	(6,7)	10	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	6	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	-17	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	-9	37*
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	0	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	1	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	3	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	-7	34*
10	(6,7), (10,11)	(6,9)	(5,9)	(9,12)	7	41

V tabulce 2 jsou odebíraná spojení tabu-aktivní po 2 iterace, přidávaná spojení pak po 1 iteraci. Počet iterací, při kterých jsou spojení tabu-aktivní, nazýváme Tabu Tenure (Tabu

držení). Pohyb je tabu, jestliže je přidávané nebo odebírané spojení tabu-aktivní. V každé iteraci zkoumáme všechny sousední body a vždy vybereme ten nejlepší, který není tabu.

Sloupec tabu držení (Net tenure) 1 a 2 – uvádí počet iterací, kdy spoje zůstanou tabu-aktivní (včetně aktuální iterace). Tyto pohyby jsou klasifikovány jako tabu. Z tabulky 2 je zřejmé, že tabu-aktivní třídění mění okolí tak, že se celková váha nejen zmenšuje, ale v mnoha případech toleruje zvyšování celkové váhy. Z tabulky 2 je zřejmé, že tyto tahy vedou k lepším řešením než bylo to počáteční. Nejlepší řešení (označené hvězdičkou) se v průběhu iterací postupně zlepšuje. V iteraci č. 6 došlo k tahu, kde celková váha zůstala nezměněna, tj. hodnota pohybu byla nula. Při takovýchto tazích může být systém upraven zvláštním nastavením tak, aby nedošlo k zacyklení. Na obrázku 5 je graficky znázorněna trajektorie tohoto příkladu.



Obr. 5 Trajektorie hledání [2]

V příkladu bychom mohli pokračovat dalšími iteracemi, ovšem pro názornou ukázku použití krátkodobé paměti je dostačující tento. Jeden ze způsobů, jak využít TS je právě pravidelné přerušení postupu, obzvlášť klesne-li míra nově nalezených řešení klesne pod upřednostňovanou úroveň, a dojde ke znovuspuštění procesu, který vytváří novou sekvenci řešení. Procedura restartu může být založena na náhodě, ovšem TS využívá také sofistikovanějších forem popsaných v následující kapitole.

7 Paměť kritické události

Paměť kritické události v zakázaném prohledávání monitoruje během hledání výskyt určité kritické události a stanovuje paměť, která tyto přehledy ukládá. Pro kritickou oblast v tomto příkladu, kde usilujeme o vytváření nových začínajících řešení, je významná tvorba předchozích začínajících řešení. To tedy znamená, že pokud použijeme vícekrát proceduru restartu, kroky všech vytvořených řešení budou tímto procesem ohodnoceny a uloženy do paměti. Základem nových řešení mohou být odlišnosti v řešeních předchozích. Různé stupně odchylek reprezentují různé úrovně obměn. Jedna z možností je použít všeobecný přístup, který považuje předešlé kompletní vytvořené řešení za kritickou událost. Seskupením takových výsledků pamětí kritické události a separací menšího počtu řešení nám udává základ k bodu nového opakování.

Pro náš příklad budeme vycházet z předpokladu, že kritické události jsou složeny z počátečního řešení a také z řešení, které představuje lokální TS optimum (tj řešení, jehož funkční hodnota je lepší - nebo ne horší, než řešení předcházející a následující). Z našeho příkladu se jedná o čtyři řešení: počáteční, z iterace 5,6 a 9 s váhou 40, 37, 37 a 34. Protože řešení v 9. iteraci se zdá být optimem, budeme řešit proceduru restartu před tímto nalezeným řešením. Jako výsledek tedy bude řešení počáteční a z 5. a 6. iterace. Poté upravíme tato tři řešení spojením do podstromu (frekvenční pamět vylepší tyto údaje spočítáním kolikrát se každé spojení objeví v kritických řešeních a povoluje zahrnutí dalších jakostních faktorů).

K provedení procedury restartu pokutujeme spojení tohoto podstromu. Obvykle je lepší pokutovat v prvotních krocích a postupně slevovat s rostoucím počtem kroků. Účel je zřejmý – aby takto pokutovaná spojení nebyla (resp. byla minimálně) použita v dalších řešeních.

Poté znovu spustíme metodu vybrání prvního spojení, které má minimální váhu a není v pokutovaném podstromu.

8 Tabu seznam

Všeobecně je aktuální paměť řešena vytvoření jednoho nebo několika tabu seznamů, které zaznamenávají tabu-aktivní atributy a rozpoznávají jejich aktuální stav. Velikost Tabu seznamu se může měnit v závislosti na typu nebo kombinaci atributů, či časovém intervalu nebo stupni hledání. Takto proměnné držení poskytuje dynamický a silný způsob hledání.

Volba vhodného druhu tabu seznamů závisí na souvislostech. Ačkoli neexistuje žádný unifikovaný typ seznamu pro všechny aplikace, lze vyjádřit některé hlavní rysy. Jestliže je paměťový prostor dostatečný k uchování jedné informace pro každý řešený atribut k určení tabu aktivačního pravidla, je obvyklé zaznamenávat číslo iterace, které identifikují, kdy má status tabu-aktivní. Typicky se používá pro testování tabu statutu pohybu v reálném čase. Potřebný paměťový prostor závisí na atributech a na velikosti okolí.

8.1 Určení velikosti Tabu seznamu

Pro určení velikosti Tabu seznamu máme dle[9] několik možností:

1) statickou pro všechny iterace
2) dynamickou v závislosti na počtu parametrů
3) náhodným výběrem z intervalu

V závislosti na velikosti úlohy a v kombinaci s jistými druhy atributů nemusí být tato paměť použitelná. Uchovávání jednoho druhu informace pro každý atribut se stává neatraktivní ve chvíli, kdy se rozsah úlohy příliš zvětšuje, anebo se jedná o složené atributy. Nicméně žádné jednotné pravidlo k získání efektivního držení nebylo zkonstruováno.

Efektivní tabu držení a tabu aktivační pravidla mohou být pro daný druh úloh stanoveny drobným experimentováním. Tabu držení, která jsou příliš malá, se často vyznačují pravidelným opakováním hodnot funkce nebo jinými funkčními ukazateli, které jsou schopny rozpoznat výskyt cyklování. Držení, která jsou velká, lze rozpoznat výsledným zhoršením nalezených řešení během přijatelného časového období. Někde mezi těmito „mantinely“ existuje velká řada velikostí Tabu seznamu, která poskytují dobrý výkon.[10]

Jednou z metod získání hodnoty velikosti Tabu seznamu je výběr z hodnot získaných z různých iterací. Obecně platí, že krátké tabu držení povoluje zkoumání řešení „blízko“ lokálního optima, zatímco dlouhá držení mohou pomoci „uvolnit se“ z okolí lokálního optima[2]. Tyto funkce se nazývají intenzifikace a diverzifikace. Proměnná hodnota tabu držení poskytuje během hledání způsob, jak ovlivnit rovnováhu mezi těsným zkoumáním v jedné oblasti a posunem do jiných částí prostoru. V situacích, kde je okolí relativně malé nebo kde hodnota délky tabu seznamu je dost velká, není vyloučeno, že veškeré tahy v iteracích jsou hodnoceny jako tabu. V tomto případě je použito základní aspirační pravidlo aspiration-by-default k povolení tahu, který je „nejméně tabu“. Pokud je tabu status přenesen do upraveného hodnocení pokutami a podněty, pak je samozřejmě pravidlo aspiration-by-

default řízeno automaticky s tím, že není třeba kontrolovat možnost, že všechny pohyby jsou tabu. Realizaci dynamických tabu seznamů můžeme rozdělit na náhodné a systematické.

8.2 Náhodné dynamické držení

Náhodné dynamické tabu držení existují ve dvou formách. Obě tyto formy využívají rozsah definovány parametry t_{min} a t_{max}. Hodnota tabu držení je náhodně vybrána z tohoto rozsahu s rovnoměrnou pravděpodobností. V prvním případě je vybrané držení udržované konstantou pro α t_{max} iterací a další jsou vybírány stejným způsobem. Druhý způsob vybírá nové *t* pro každý atribut, který se stává tabu v dané iteraci. První forma vyžaduje více evidování než druhá, protože si musí pamatovat poslední čas (číslo iterace), kdy bylo tabu držení upravené.

8.3 Systematické náhodné držení

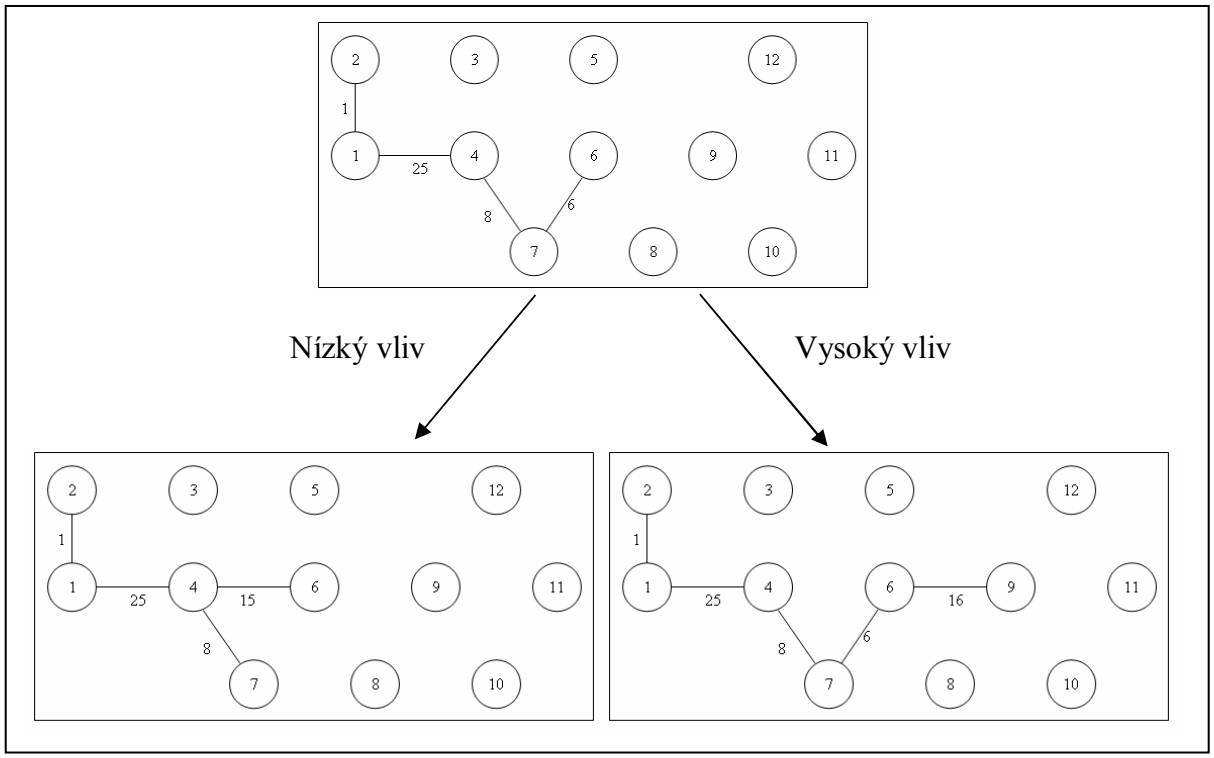
Dynamické držení založené na náhodném návrhu je lákavé pro své snadné provedení. Avšak spoléhat se na nahodilost nemusí být nejlepší volba, když jsou dostupné určité ucelené údaje. Jednoduchý způsob systematického držení spočívá ve vytvoření sekvence hodnot z rozsahu definovaném t_{min} a t_{max}. Tento rozsah je použit ke změně *t* tak, že jeho hodnotu střídavě zvyšujeme a snižujeme. Sekvence může být opakována mnohokrát až do konce hledání, kde její další obměna spočívá v progresivním posunování, anebo obrácením pořadí.

Dobře navržené adaptivní systémy mohou významně redukovat, či dokonce odstranit potřebu zjišťovat nejlepší rozsah hodnot úvodní kontrolou.

Tyto základní možnosti typicky poskytují dobrý nástup k provedení zakázaného prohledávání. Ve skutečnosti pro většinu počátečních zkoumání, jsou použity pouze nejjednodušší varianty těchto metod.

8.4 Aspirační kritérium a oblastní závislosti

Aspirační kritérium slouží v zakázaném prohledávání k tomu, aby určil, kdy mohou být tabu pravidla potlačena. Prvotní použití, pouze jako jednoduchý typ aspiračního kritéria, sestává z odstranění tabu ohodnocení na ukázkovém tahu, kdy tento tah přináší lepší řešení než doposud získané. Toto kritérium zůstává široce užívané. Základní pro jedno z těchto kritérií vyvstává se zavedením pojmu „ovlivňování“ (influence), které měří stupeň vyvolaných změn ve struktuře řešení[2].



Obr. 6 Stupeň vlivu na dvou tazích [2]

Z obrázku 6 jsou zřejmé rozdíly ve váhách jednotlivých způsobů pohybu. Ovšem rozdíly v celkové váze nejsou tou nejpodstatnější částí. U příkladu s malým vlivem vytváří pohyb řešení, které má stejnou sadu uzlů jako řešení aktuální, zatímco pohyb s vysokým vlivem obsahuje tah, kde můžeme prozkoumávat z nového uzlu nová spojení. Úloha vysokých vlivů je důležitá zejména pro úniky z lokálního optima, protože u série tahů, která vytváří pouze omezené změny, je malá pravděpodobnost významného zlepšení.

Aspirace lze rozdělit podle provedení do dvou kategorií: aspirace tahu a aspirace atributu. Aspirace tahu ruší tabu pohyb, aspirace atributu ruší atribut tabu-aktivního statutu. Tabu tah smí být změněn pouze v případě, že tabu aktivační pravidlo je spuštěné na více než jednom atributu. Což v případě zrušení tabu na statutu ruší také tabu na tahu.

9 List kandidátů

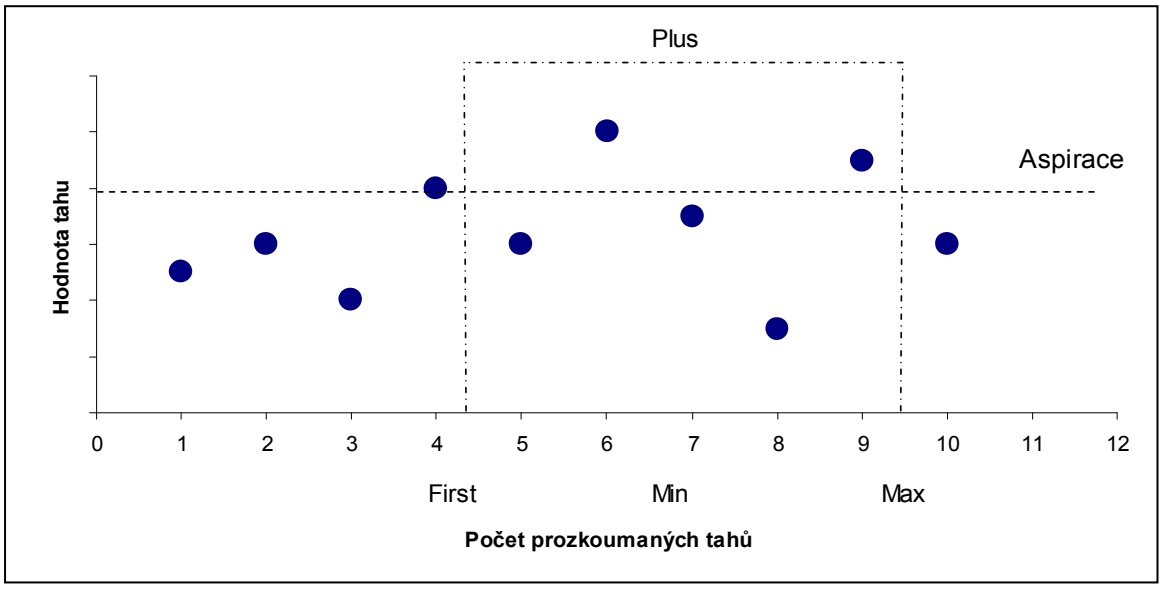
Důležitý aspekt zakázaného prohledávání je evidentně ve výběru pravidel, která hledají nejlepší dostupný pohyb získaný s patřičným úsilím. Správné určení pravidel a hodnocení vhodných kandidátu je zásadní pro vyhledávací proces. Účelem je separovat oblasti okolí obsahující tahy s vhodnými vlastnosti a umístění těchto tahů na list kandidátů pro další prozkoumání. Efektivitu strategie získávání listů kandidátů lze rozšířit použitím

tzv. přímé paměti, která aktualizuje hodnocení pohybů od jedné iterace k další. Pokud jsou tyto aktualizace vhodně navržené, mohou znatelně snížit námahu k nalezení nejlepšího tahu.

Listy kandidátů mohou být konstruovány v kontextu s příslušnými pravidly nebo pomocí všeobecných pravidel. U takových pravidel musí být kladen důraz na to, aby efektivita získávání kandidátů nebyla měřena v rámci jedné iterace, tzn. že vhodnější měřítko výkonu pro daný list kandidátů je kvalita nejlepšího řešení získaného během stanoveného času. Pokud se kvalita nejlepšího řešení během této časové lhůty nezlepší, vyvozujeme, že tento typ strategie není příliš účinný. Několik všeobecných strategií je popsáno níže.

9.1 Aspirační plus

Strategie aspiračního plus stanovuje hranici pro kvalitu pohybu vycházející z historie nalezeného ideálu. Tato procedura pracuje se zkoumanými tahy, dokud nenalezne takový, který splňuje nastavený práh. Při dosažení tohoto bodu jsou prozkoumány dodatečné tahy, které splňují kritéria PLUS, a poté je vybrán nejlepší pohyb. Pro zajištění, že nebylo prohledáno ani příliš málo, ani příliš hodně pohybů, je zavedeno pravidlo, které určuje minimální a maximální počet pohybů, které budou prozkoumány – pravidlo Min a Max. Stanovení těchto hodnot ukázaných na obrázku 7 je následující. Označíme si počet pohybů, které byly potřeba k nalezení prvního tahu, který splňuje limit – hodnota FIRST, další pohyb pak označíme jako PLUS. Pokud nebyly stanoveny hodnoty Min a Max, pak celkový počet kroků bude FIRST + PLUS. Nicméně, pokud FIRST + PLUS < Min, pak platí hodnota min, jestliže FIRST + PLUS > Max, pak platí Max.



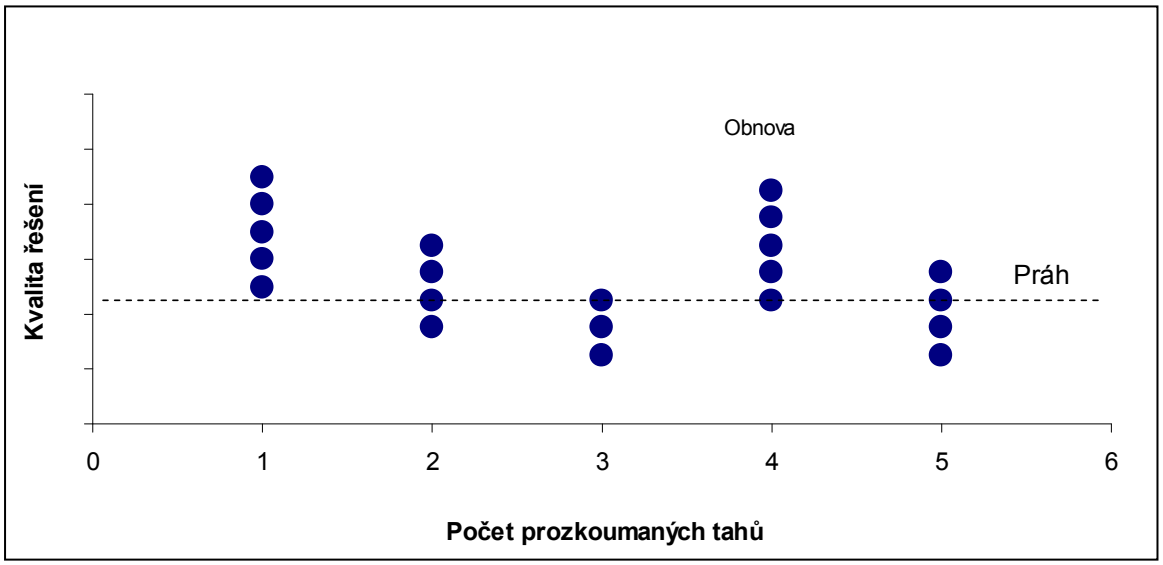
Obr. 7 Aspirační plus [2]

V obr 7. 4. tah vyhovuje stanovenému prahu aspirace a je označen za FIRST. Za hodnotu PLUS byla zvolena 5, tzn., že celkem bude zkoumáno 9 tahů. Nejméně by bylo zkoumáno 7 tahů, a maximální hranice je nastavena na 11. Obě podmínky FIRST + PLUS pro Min x Max byly splněny, proto zde pracujeme s hodnotou 9 tahů. Nejlepší tah nalezený v tomto příkladu byl 6.

„Aspirační hranice“ může být během hledání dynamicky upravována, stejně jako hodnoty Min, Max nebo funkce počtu pohybu. Je zřejmé, že čím delší bude sekvence, tím více se bude řešení zlepšovat.

9.2 Elitní list kandidátů

Další metoda je založena na sestavení tzv. hlavního listu ukázaném na obrázku 8, kde zkoumá všechny tahy (nebo velké množství tahů), vybráním k nejlepších pohybů, kde k je parametr procesu. Spodní hranice hlavního listu určuje práh kvality a iterace probíhají do té doby, dokud nejlepší prvek z aktuálního hlavního listu „nespadne“ pod tuto úroveň, nebo neproběhne stanovené množství iterací. Pak je hlavní list zkonstruovaný a proces se opakuje.

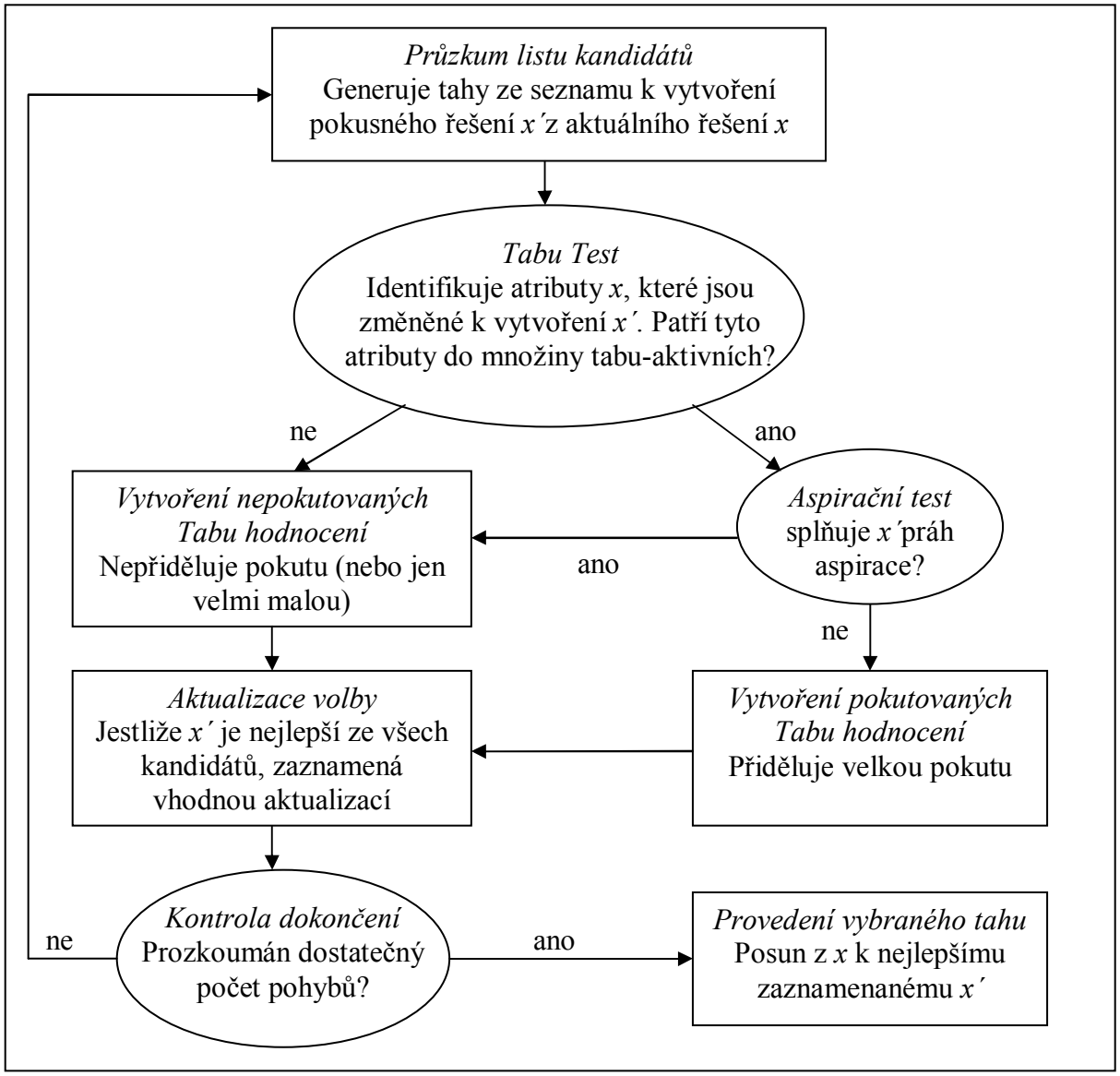


Obr. 8 Elitní list kandidátů [2]

Tato technika je podnícena předpokladem, že tah, který nebyl proveden na aktuální iteraci, může být ještě dobrý pro nějaké další iterace. Po provedené iteraci může být vlastnost zaznamenaného pohybu transformována, hypoteticky by v tomto případě mělo platit, že vhodná část těchto transformovaných tahů zdědí atraktivní vlastnosti od jejich předchůdců. Technika elitního listu kandidátů může být vhodně rozšířena metodou aspiračního plus,

dovolující prozkoumání dodatečného množství tahů mimo hlavní seznam pro záměnu prvků hlavního seznamu.

10 Operace krátkodobé paměti



Obr. 9 Operace krátkodobé paměti [2]

Obrázek 9 popisuje postup, jakým způsobem pracuje metoda Tabu search s krátkodobou pamětí. Znázornění pokut vyjadřuje hraniční rozdíly mezi velkými, anebo velmi malými - buď tabu status přináší velmi špatná hodnocení, anebo slouží zejména k rozbití vazeb mezi řešeními s nejvyšším hodnocením. Pokud všechny aktuálně dostupné tahy vedoucí k řešení jsou tabu, pokuty mají za následek výběr „nejméně tabu“ řešení. Pořadí

testování tabu a testu aspirace může být obrácené, a to v případě, kdy není splněna podmínka aspiračního prahu. Tabu hodnocení může být upraveno vytvářením podnětů založených na aspiračním stupni, jako je upravování vytvářením trestů založených na tabu statutu. V tomto smyslu aspirační podmínky a tabu podmínky mohou být chápány jako zrcadlově otočené.

11 Kritéria ukončení

Pokud bychom nenastavili kritéria, kdy se proces ukončí, mohli bychom teoreticky pokračovat navždy. Tedy v případě, že neznáme předem námi hledaný výsledek. Musíme stanovit, jakým způsobem stanovíme ukončení běhu prohledávání.

Nejběžněji používané metody zastavení jsou [4]:

a) po určitém počtu opakování (nebo počtu jednotek strojového času)

b) po určitém počtu iterací, kdy nedošlo ke zlepšení

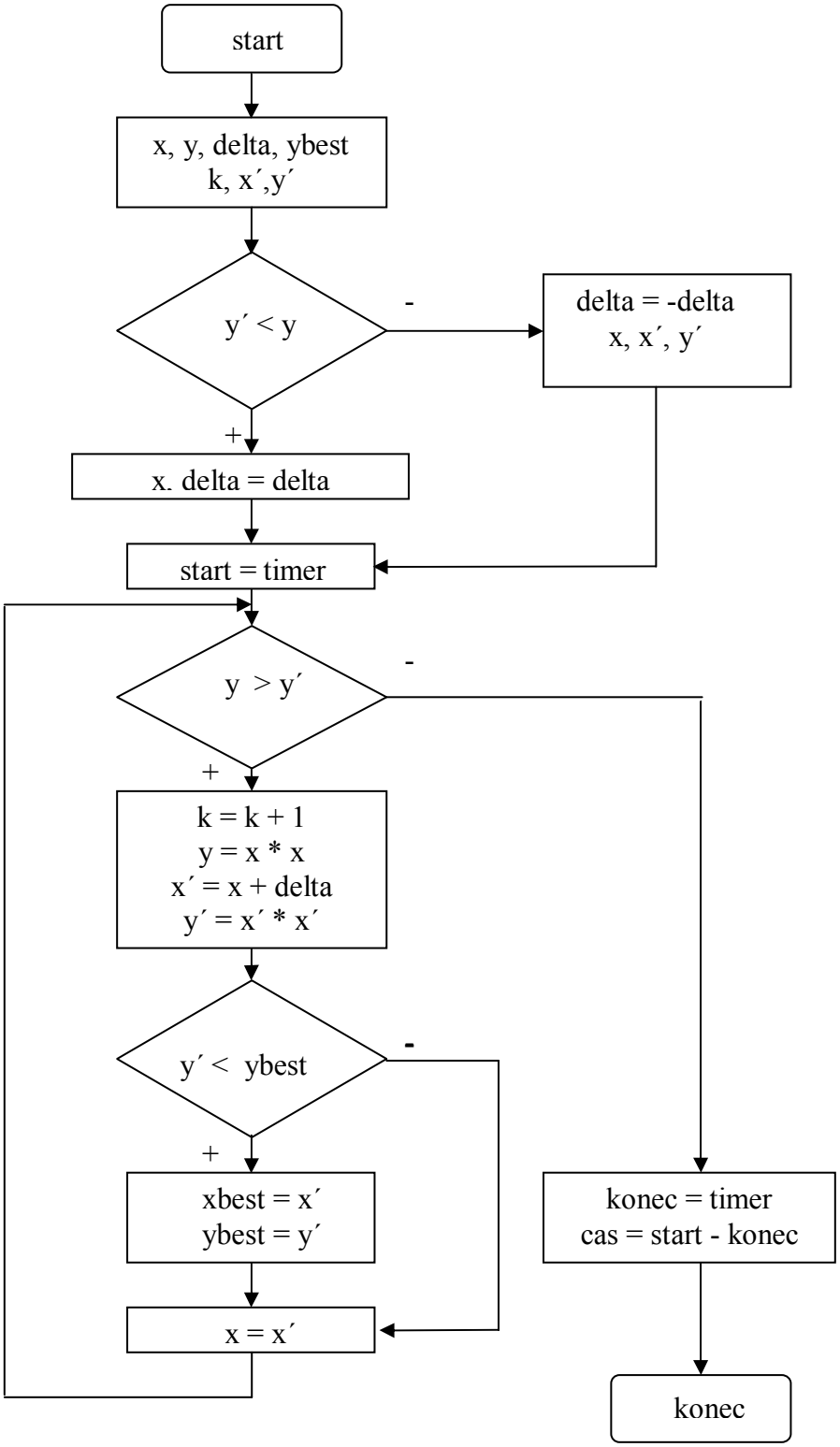
c) když se hodnota dostane pod námi předem určený práh

12 Vyhledání lokálního minima

Jak již bylo uvedeno v úvodu, metoda TS se dá považovat za „nadstavbu“ vyhledávání lokálního minima – horolezeckého algoritmu popsaného na obrázku 10. V této kapitole bude pomocí tohoto algoritmu osvětleno několik základních proměnných, které budou dále použity v algoritmech pro zakázané prohledávání.

Nevýhodou horolezeckého algoritmu pro vyhledávání globálního minima je, že v průběhu provádění uvízne v lokálním minimu, tedy dojde k jeho zacyklení. Ovšem při stanovení podmínky, že se ukončí ve chvíli, kdy funkční hodnota nově nalezeného prvku je větší než aktuální, dojde k zastavení algoritmu. Tento typ je předveden na kvadratické funkci $f(x) = x^2$, kde se lokální minimum rovná minimu globálnímu.

12.1 Algoritmus vyhledání lokálního minima



Obr. 10 Algoritmus vyhledání lokálního minima

Na začátku algoritmu znázorněného na obrázku 10 si deklarujeme základní proměnné, kde:

- x – funkční hodnota
- y – hodnota funkce x
- delta – přírůstek k funkční hodnotě (který budeme na základě sestupné funkce buď přičítat, nebo odečítat)
- ybest – nejlepší získaná hodnota y
- k – počet kroků (iterací)
- x´, y´ - funkční hodnota změněná o deltu, resp. hodnota funkce y´
- cas – čas běhu algoritmu

Poté si zvolíme náhodně x z předem stanoveného intervalu a pomocí funkce určíme, jakým směrem se na ose x budeme pohybovat. Podmínka, kde modifikovaná f(x´) je menší než f(x), znamená, že přírůstek delta budeme přičítat, tzn. že se nacházíme vlevo od lokálního minima. Tato funkce bude použita i v následující kapitole algoritmu TS.

```
Private Sub local_minimum()
Dim time, timestart, timeend As Integer
Dim k As Long
Dim x, y, xbest, ybest, x´, y´ As Double
Dim intod, intdo As Integer
Dim fce, delta As Variant
Dim start, konec, cas As Variant
Dim db As Database
Dim rst As Recordset

Set db = CurrentDb
Set rst = db.OpenRecordset("tbl_vysledky_xna2", dbOpenDynaset)

delta = 0.00002
x = 5.2
y = x * x
ybest = y
x´ = x + delta
y´ = x´ * x´
k = 0

If y´ < y Then
 delta = delta
 x = 5.2
 Else
  delta = -delta
  x = 5.2
  x´ = x + delta
  y´ = x´ * x´
```

```
End If

start = Timer

Do While y > y´
  k = k + 1
  y = x * x
  x´ = x + delta
  y´ = x´ * x´
   If y´ < ybest Then
    xbest = x´
    ybest = y´
   End If
  x = x´
Loop
konec = Timer
cas = konec - start

With rst
 .AddNew
 !k = k
 !ybest = ybest
 !cas = cas
 !delta = delta
 .Update
End With

MsgBox "k =" & k & ", ybest=" & ybest & ", cas = " & cas,
vbOKOnly
End Sub
```

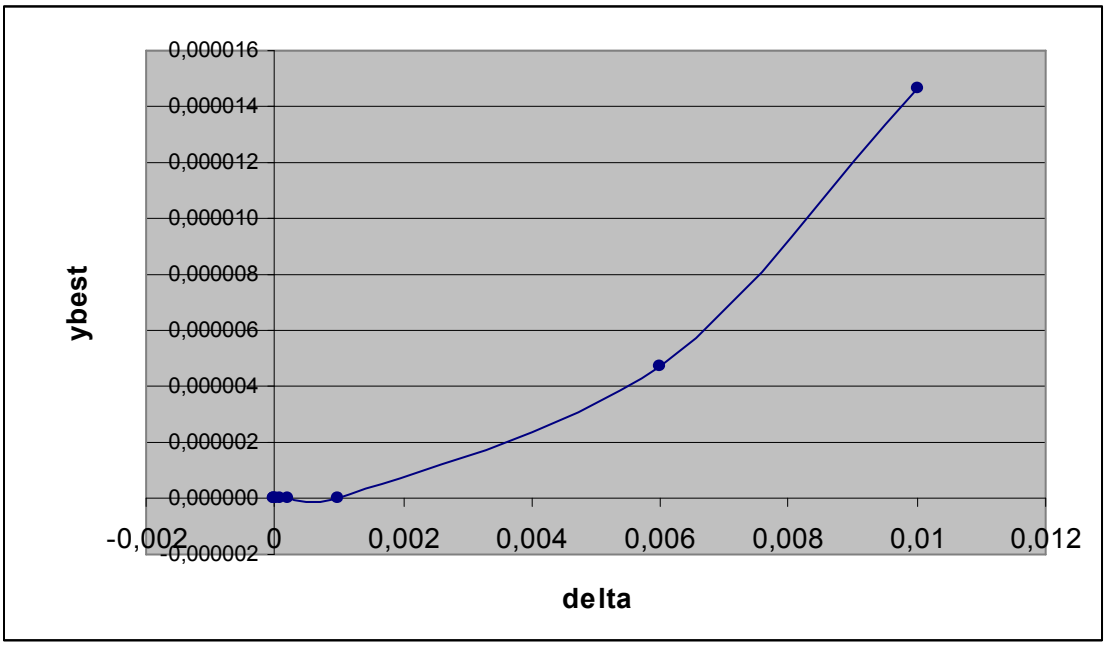
V tabulce 3 jsou uvedeny výpočty získané při změnách hodnot delta-

Tab. 3 Výpočty proměnných funkce x^2

delta	ybest	xbest	cas	k
0,01	0,000014668900000000000	-0,00383000000007808000	0	577
0,006	0,000004708900000000000	0,00216999999999696380000	0	962
0,001	0,000000028900000000000	0,00017000000259678000	0	5765
0,0002	0,000000008999998000000	-0,00002999999668530090	0,0156	28820
0,0001	0,000000009000000000000	-0,00003000000009357050	0,0312	57639
0,00002	0,000000000999995900000	-0,00000999997917450418	0,0937	288192
0,0000002	0,000000000000000049223	-0,00000000221857218713	10,375	28819150
0,00000002	0,000000000000000029075	0,00000001705167198466	102,25	288191500

Pro tyto výpočty byla zvolena konstantní hodnota x = -5.76383, jelikož při náhodném určení by údaje byly zkreslené. Z tabulky a obrázku 11 je patrné, že i mírná změna hodnoty delta silně ovlivní sledované proměnné. Existuje zde přímá úměrnost mezi nejlepší hodnotou

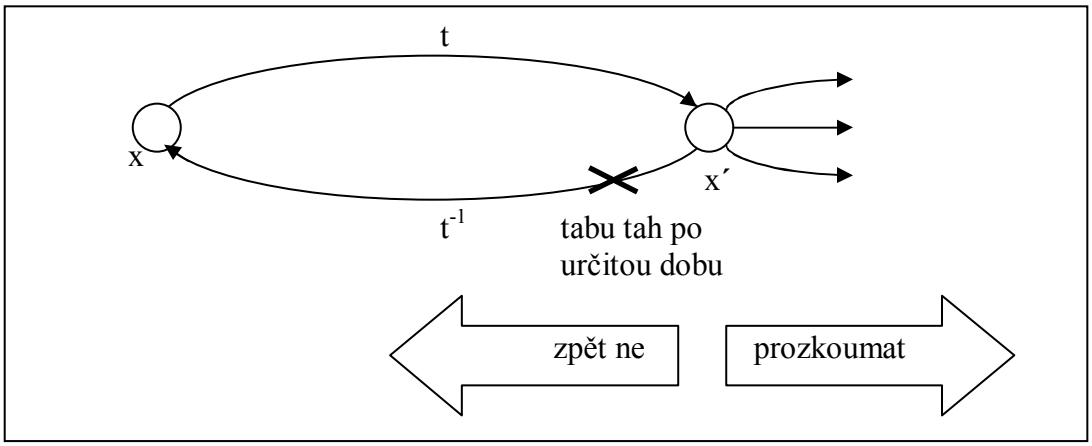
(*ybest*) a počtem kroků, resp. časem k jeho získání. Neméně důležitým aspektem je také interval, v kterém je x náhodně vybíráno, což by mělo být při volbě delty bráno v potaz.



Obr. 11 Graf závislosti ybest na proměnné delta

13 TS s krátkodobou pamětí

V případě algoritmu TS s krátkodobou pamětí již nestačí cyklus s podmínkou na začátku, ale musí být použít cyklus s určitým počtem opakování. Tento cyklus může být omezen například na určitý počet kroků nebo na stanovený čas běhu funkce. Zde se vrátíme k pojmu diverzifikace popsaném na obrázku 12, kde nám následující funkce zabraňuje příliš časnému návratu k výchozímu stavu.



Obr. 12 Diverzifikace [11]

13.1 Realizace Tabu seznamu

Bude zaveden datový typ fronta, do které budou zapisovány prohledané tahy označené jako Tabu, tj. dojde k vytvoření Tabu seznamu. Tabu seznam je realizován jako matice n x n z celočíselných hodnot [12]. Datový typ fronta bude sloužit k dočasnému uložení prvků, kde se prvek, který se ukládá dříve, také dříve odebere. Fronta je označována jako „FIFO“ (z angličtiny „first-in first-out“). V této úloze je realizována jako vektor s pevným počtem prvků, který se bude v průběhu výpočtů manuálně měnit. Právě velikost tohoto vektoru je ekvivalentem v TS jako velikost tabu seznamu, nebo tabu lhůty**.** Jedná se o výrazy přeložené z angličtiny, přesný překlad by byl tabu držení – v angličtině Tabu Tenure popsané v kapitole 8.

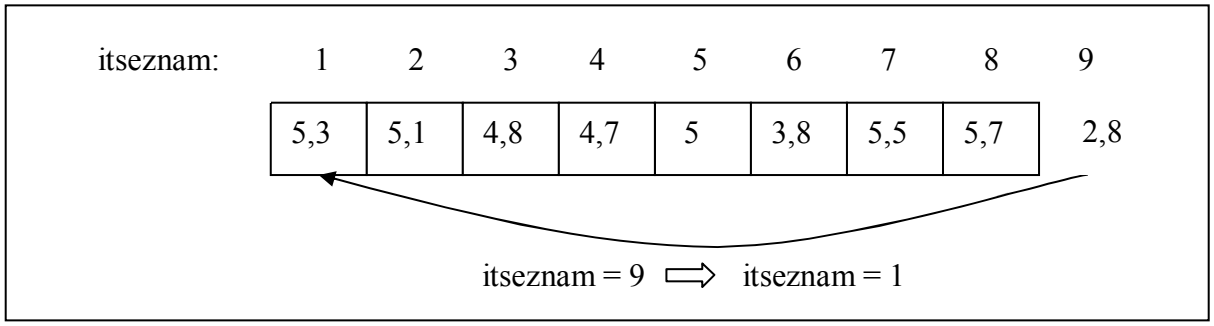
13.1.1 Vytvoření Tabu seznamu

Pro vytvoření tabu seznamu je zavedena proměnná pro délku tabu seznamu *delkatseznam* a pomocnou hodnotu *itseznam*. Vektor deklarovaný jako tabuseznam(1 to delkatseznam) As Datový typ**.** Následuje kód funkce na přidání prvku do tabuseznamu:

```
Function tseznampridej()
Dim itseznam, delkatseznam As Long
Dim tabuseznam(1 To delkatseznam) As double

If itseznam > delkatseznam Then itseznam = 1
 tabuseznam(itseznam) = x´
 itseznam = itseznam + 1
Next
End Function
```

Podmínka, ve které se porovnává délka tabu seznamu s pomocnou hodnotou, slouží k přesunu na první prvek seznamu v případě, že předchozí hodnota byla zapsána do posledního prvku v seznamu. Obrázek 13 popisuje tuto situaci pro délku tabu seznamu 8, kde se snažíme zapsat devátý prvek.



Obr. 13 Realizace fronty v tabu seznamu

13.1.2 Hledání v Tabu seznamu

V další fázi algoritmu zakázaného prohledávání, když už je vytvořen tabu seznam a lze do něj zapisovat, budou nově získané prvky z okolí porovnávány s prvky v tabu seznamu, resp. transformace nově získané srovnávat s těmi označenými jako tabu.

Tato funkce je realizována For cyklem ukázaným v následujícím kódu:

```
Function jevtseznamu() As Boolean
 Dim x As Double
 Dim itseznam As Long
 x = prvek(transformace)
 For itseznam = 1 To delkatseznam
  If x = tabuseznam(itseznam) Then
   jevtseznamu = True
   Exit Function
  End If
 Next
End Function
```

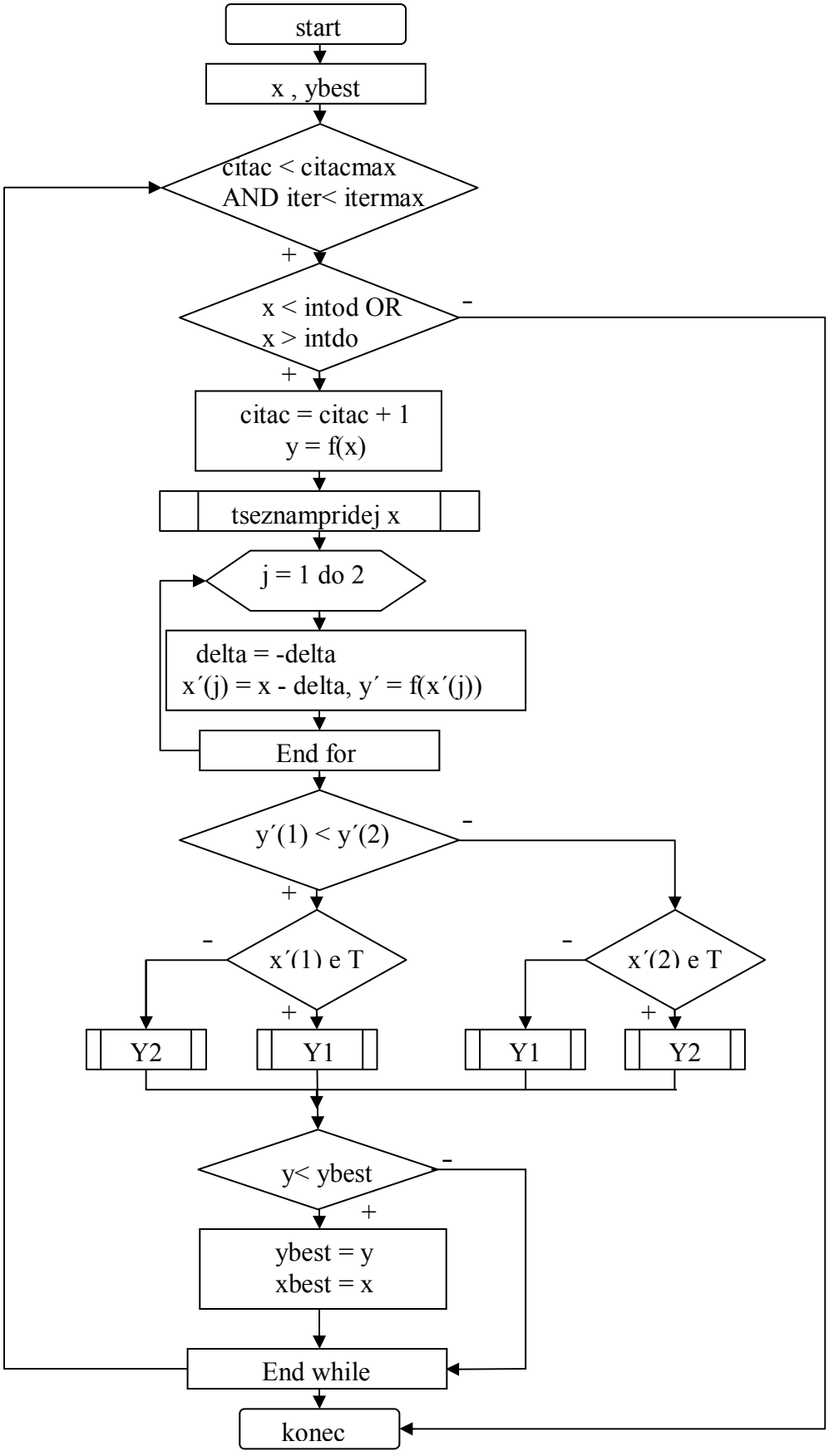
Je porovnáván nově vygenerovaný prvek, který je kandidátem na nové okolí, s prvky v tabu seznamu. Obě tyto funkce budou implementovány do algoritmu zakázaného prohledávání s využitím krátkodobé paměti.

Algoritmus TS Short Term Memory

Základní zápis algoritmu TS uvedený na obrázku 14 je definován [13] následujícími kroky:

1) vygeneruj náhodně x a označ ho jako aktuální
2) vyhodnoť toto x a hodnotu zaznamenej jako nejlepší
3) ulož x do Tabu seznamu
4) vytvoř N počet možných tahů z aktuálního bodu a vyhodnoť je
5) porovnej seřazené tahy a pokud :
    a) má tah nižší funkční hodnotu, akceptuj to
    b) tah není tabu, akceptuj to
    c) neexistují přípustné tahy, ukonči algoritmus
6) pokud nastala situace a + b, pokračuj v případě, že podmínka „maximálních počet přípustných tahů“ nebyla překročena, inkrementuj tuto hodnotu a pokračuj bodem 2. V opačném případě, kdy je funkční hodnota menší, zapiš ji jako lokální minimum a porovnej s minimem globálním.

13.2 Algoritmus Tabu search



Obr. 14 Algoritmus TS s krátkodobou pamětí

```
x = randomize
ybest = f(x)

start = Timer
Do While (citac < citacmax) And iter < itermax
 citac = citac + 1

If x < intod Or x > intdo Then GoTo konec

y = f(x)
tseznampridej

For j = 1 To 2
 delta = -delta
 x´(j) = x - delta
 y´(j) = f(x´(j))
Next

If y´(1) < y´(2) Then
  If jevtx1 = False Then
   y1
   Else
    y2
  End If
  Else
   If jevtx2 = False Then
    y2
    Else
     y1
   End If
End If

If y < ybest Then
 ybest = y
 xbest = x
End If

Loop
konec:
konec = Timer
cas = konec - start
```

13.2.1 Funkce randomize

V úvodu je třeba nastavit proměnné ybest, čítač, maximální počet přípustných iterací a také hodnotu delta. Hodnota x je určena funkcí *randomize* v kombinaci s funkcí *Rnd* podle zvoleného intervalu. Jak už bylo uvedeno, hodnotu delta je třeba nastavit v závislosti

na velikosti intervalu, ve kterém budeme globální minimum hledat. Následující ukázka kódu představuje náhodný výběr hodnoty v intervalu <-5.12; 5.12>

```
Randomize
x = Int(512 * Rnd)
x = x * 0.01
plusminus = (Int(2 * Rnd))
If plusminus = 1 Then x = -x
```

13.2.2 Určení proměnných

Algoritmus pracuje, dokud není splněna podmínka dosažení počtu průchodů, nebo nedošlo k překročení povoleného množství akceptovatelných iterací. U první veličiny bude nastavena hodnota v závislosti na velikosti delta a velikosti intervalu, ve kterém bude optimum hledáno.

U druhé veličiny je hodnota opět závislá na velikosti delta. Hodnota itermax určuje počet iterací, u kterých je akceptovatelná zvýšená funkční hodnota. Optimální velikost bude popsána v následující kapitole na Rastriginově funkci.

Jestliže je podmínka splněna, prověříme, zda se proměnná x nachází v námi zadaném intervalu. Když splněna není, znamená to, že jsme se dostali mimo stanovený interval, a proces se ukončí.

Dále zjistíme funkční hodnotu aktuálního x a tento prvek pomocí funkce tseznampridej umístíme do Tabu seznamu.

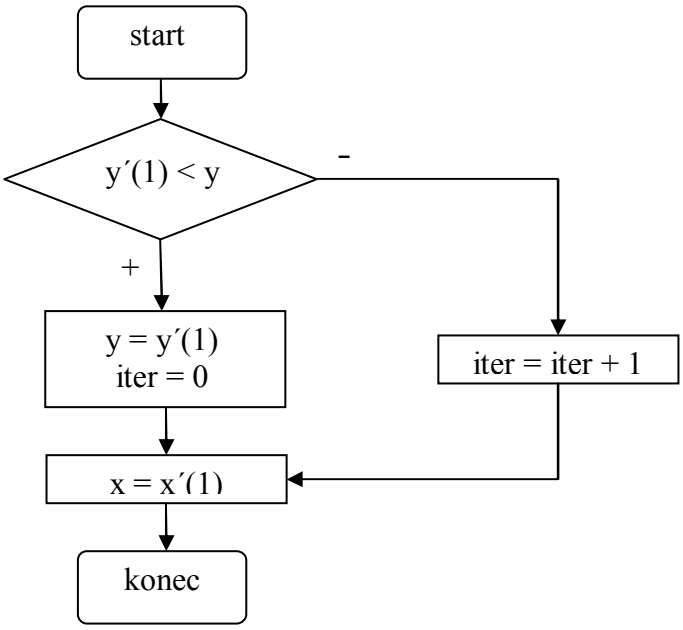
13.2.3 Porovnání účelových funkcí

For cyklus slouží k získání okolních tahů, v případě jednorozměrné funkce budeme od aktuální hodnoty x odečítat, resp. přičítat hodnotu delta. Tímto postupem získáme dva možné tahy od aktuálního x. X´(j), stejně tak jako y´(j), které nám udávají velikost funkčních hodnot těchto veličin, jsou realizovány pomocí datového typu fronta deklarovaných v globálních proměnných z důvodů používání funkcí na prohledávání v Tabu seznamu.

```
Dim x´(1 To 2) As Double
Dim y´(1 To 2) As Double
```

Porovnáním hodnot funkce y´(1) a y´(2) určujeme, který aspirant z možných tahů bude zvolen za aktuální. V případě vyhledání minima vybereme samozřejmě ten prvek, který má

nižší funkční hodnotu. Podstatnou roli zde bude hrát také fakt, zda tah patří do množiny zakázaného seznamu. Tedy v situaci, že je funkční hodnota x´(1) menší než funkční hodnota x ´(2), budeme testovat podle výše zmíněné funkce jevtseznamu prvek x´(1). Pokud se tedy tato hodnota v Tabu seznamu nenachází, provedeme funkci Y1.



Obr. 15 Algoritmus funkce Y1

V dvojité podmínce popsané na obrázku 15 jsme se dostali do situace, kdy byl jeden z tahů eliminován – v první fázi kvůli vyšší funkční hodnotě, ve fázi druhé – rozhodující – zjištěním přítomnosti v Tabu seznamu. Funkce Y2 je identická s Y1, jediný rozdíl je ovšem v porovnávaných veličinách. Tato funkce slouží k porovnání funkční hodnoty aktuálního prvku a prvku nového. Pokud je splněna podmínka minimalizace, nová funkční hodnota nahradí původní, což se projeví v následující podmínce, kde se porovnává s nejlepší dosud získanou hodnotou. Současně je veličina iter nastavena na nulu. Ve chvíli, kdy podmínka splněna není, proměnná iter je inkrementována o jednotku. Tato hodnota je v každém průchodu cyklu s podmínkou na začátku porovnávána s omezující proměnnou itermax. V závěru funkce se pak hodnota x´(1) stává novým výchozím řešením.

13.2.4 Zaznamenání nejlepší hodnoty

Poslední podmínkou je porovnání aktuální funkční hodnoty y s nejlepším řešením ybest. Při splnění dojde k zápisu hodnot do proměnných xbest a ybest. Celý tento proces se opakuje do té doby, dokud není splněno jedno ze stanovených omezení.

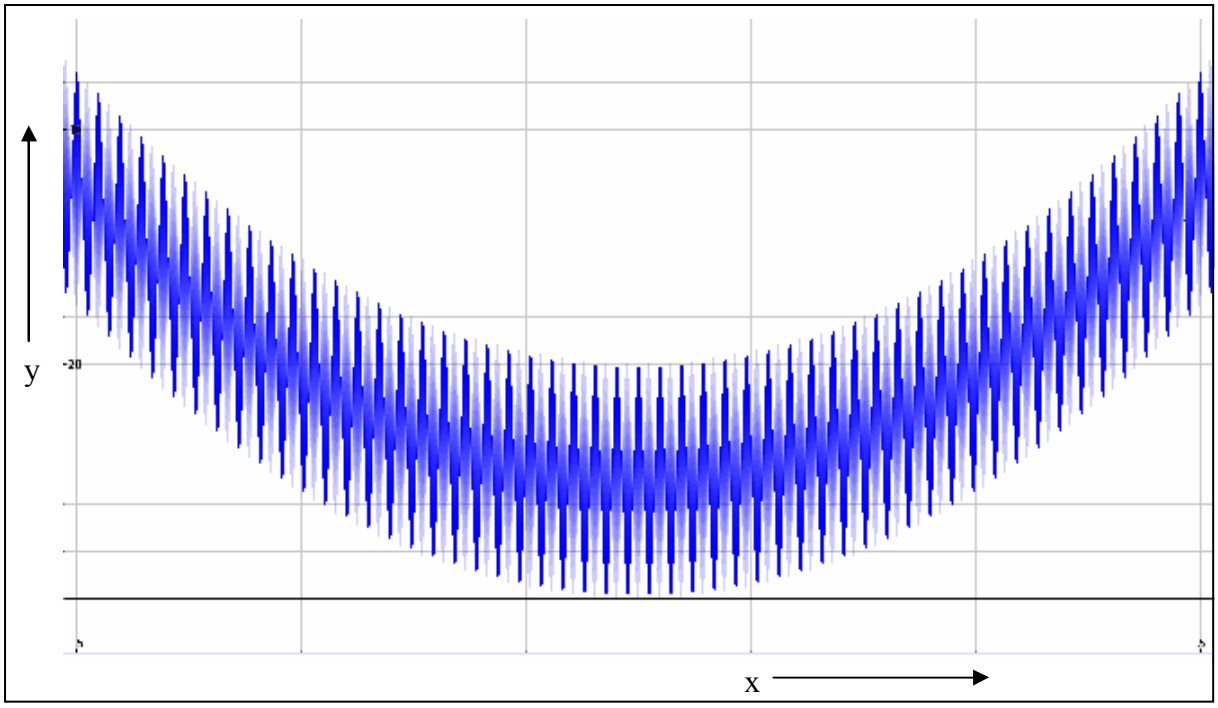
V průběhu algoritmu zaznamenáváme čas výpočtu pomocí funkce *Timer* v Accessu, kde odečítám hodnotu časovače na začátku cyklu od hodnoty na konci celého cyklu. Tuto hodnotu poté ukládáme pomocí sady záznamů *Recordset* do tabulek výsledků.

14 Rastriginova funkce v jednorozměrném prostoru

Použití algoritmu TS s krátkodobou pamětí si předvedeme pro prvotní výpočty s Rastriginovou funkcí v jednorozměrném prostoru, tzn. pro n = 1.

$$f(x) = nA + \sum_{i=1}^{n} (x_i^2 - A\cos(2\pi x_i))$$

Hodnotu A bude nastavena na 10. Kdybychom této hodnotě přiřadil nulu, jednalo by se opět o kvadratickou funkci x^2, a interval, ve kterém bude funkce prověřována, bude nastaven od -5,12 do 5,12. Její průběh je znázorněn na obr 16.
Zápis funkce do programu bude tedy vypadat takto: 10 + ((x * x) - 10 * (Cos(360 * x)))



Obr. 16 Průběh Rastriginovy funkce v 1 rozměrném prostoru

Tuto funkci podrobíme testování z důvodů zjištění optimálních parametrů. Pro začátek nastavíme hodnotu delta na 0,01.

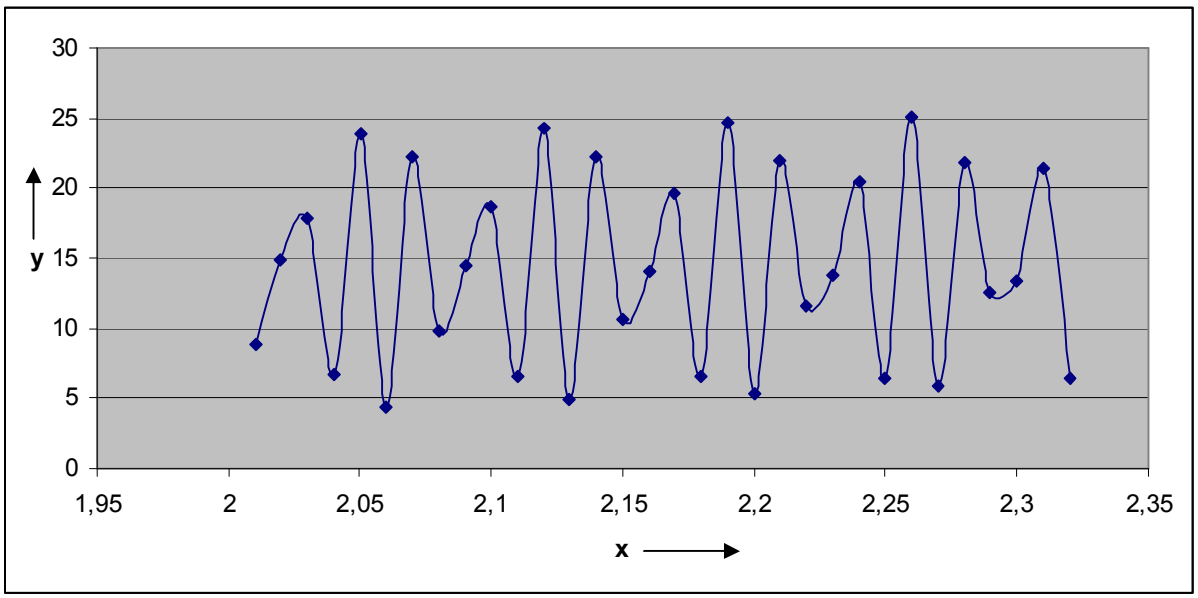
Ve výpočtech se zaměříme na parametr k, což je počet kroků, které byly provedeny k získání globálního minima, čas výpočtu, parametr x jako xbest a jeho funkční hodnotu, tedy

získané globální minimum, ybest. S touto deltou bylo provedeno celkem 71 pokusů vyhledávání, v tabulce 4 je uvedeno prvních 26.

Tab. 4 Výsledky algoritmu na Rastriginově funkci pro delta = 0,01

počáteční x	k	xbest	ybest	cas	itermax
-3,64	879	0	0	0,015625	2
-4,09	106	-4,12	17,64488	0	2
0,35	552	0	0	0	2
3,39	176	3,43	12,15589	0	2
-1,98	317	-1,99	4,028321	0	2
1,45	370	1,5	2,869843	0	2
-0,67	448	-0,76	0,718681	0	2
2,62	253	2,67	7,209883	0	2
-4,54	61	-4,59	21,0983	0	2
4,65	50	4,66	21,71615	0	2
5,01	1018	0	0	0,015625	2
2,43	272	2,46	6,588356	0	2
-5,07	8	-5,08	26,56905	0	2
0,09	506	0,14	0,109939	0,015625	2
3,22	193	3,23	11,26456	0	2
2,51	266	2,53	6,741823	0	2
0,36	551	0	0	0	2
0,83	598	0	0	0	2
2,75	790	0	0	0	2
-3,75	890	0	0	0	2
4,81	36	4,8	23,11683	0	2
-1,27	642	0	0	0	2
2,72	787	0	0	0,015625	2
1,5	665	0	0	0	2
3,56	871	0	0	0	2
4,1	925	0	0	0	2

Do algoritmu byla přidána funkce na zjištění maximálního množství akceptovatelných tahů, které mají vyšší funkční hodnotu než aktuální. Tento údaj je uveden pod sloupcem itermax.



Obr. 17 Graf průběhu Rastriginovy funkce při delta = 0,01 v okolí bodu 2,15

Z výsledků, stejně jako z grafu 17 je patrné, že maximální hodnotu itermax stačí nastavit na číslo vyšší než 2. Tím, že ve funkci náhodného generování počátečního x byly použity hodnoty s maximálně desetinnou čárkou, dosáhli jsme při výpočtech přesného globálního minima, tj. při x = 0 byla funkční hodnota y = 0. Kdyby počáteční x bylo určeno více než dvěma desetinnými místy, pak by se nejlepší výsledky pouze přibližovaly k těmto hodnotám.

V tabulce 4 jsou ovšem hodnoty xbest a ybest, které se tomuto minimu zdaleka nepřibližují. To je dáno počátečním určením, kde se rozhodujeme, jakým směrem bude prohledávání pokračovat. V situaci, kdy x bude hodnota kladná, měli bychom se v případě Rastigina posouvat v tazích směrem doleva. Ovšem, pokud je funkční hodnota tahu vpravo nižší než vlevo a tah není tabu, přijmeme tento tah jako výchozí a původní prvek je zapsán do Tabu seznamu. Tím se v průběhu algoritmu přesouváme doprava. Je ukončen až ve chvíli, kdy se dostane mimo námi stanovený interval. Výsledkem je pak globální minimum z tohoto intervalu. V tabulce 5 je uvedena statistika výsledků, které dosáhly globálního minima a těch, které dosáhly výsledků výše uvedených.

Tab. 5 Počet výsledků přibližujících se globálnímu minimu

minimum	absolutní počet	relativní počet
0	37	52,11%
>0	34	47,89%
celkem	71	100,00%

Algoritmus v 52,11% případu určil podle podmínek jít správným směrem.

Jak bylo již popsáno v kapitole o vyhledání lokálního minima na funkci x^2, velikost delty ovlivní výsledné lokální minimum. U Rastriginovy funkce je to o to složitější, neboť při zvolení vysoké hodnoty delta můžeme přeskočit interval, ve kterém se nachází skutečné minimum globální. V následující tabulce jsou uvedeny výpočty s parametrem delta = 0,01 a náhodně vygenerovaným x se třemi desetinnými místy:

Tab. 6 Výsledky TS na Rastriginově funkci, des. hodnota Random < des. hodnota delta

xpocatecni	k	xbest	ybest	cas
3,476	167	3,526	12,55499	0,015625
-1,783	695	0,167	0,028758	0
-0,208	496	-0,628	0,460059	0
-0,075	507	-0,115	0,061874	0,015625
1,557	359	1,657	3,468693	0
-0,014	516	0,436	0,261571	0,015625
3,288	845	-0,122	0,034282	0
2,177	732	0,147	0,028758	0
0,199	495	0,289	0,119186	0,015625
-2,184	296	-2,234	4,991054	0
-1,051	409	-1,231	1,834623	0
-3,208	835	0,122	0,034282	0
1,61	354	1,65	2,929359	0
-1,081	623	0,289	0,119186	0
4,418	73	4,468	19,96422	0
3,519	866	0,269	0,119186	0
-2,583	773	0,167	0,028758	0
-4,876	27	-4,886	24,4187	0
0,898	425	0,908	0,943369	0,015625
-2,717	786	-0,147	0,028758	0,015625

Přestože v počáteční fázi algoritmus rozhodne jít směrem ke skutečnému globálnímu minimu, díky hodnotě delta, jejíž počet desetinných míst je menší než hodnoty x náhodně generované, přeskočíme toto skutečné minimum a globálním minimem jsou hodnoty, které pouze inklinují k x = 0, resp y = 0. Závěr z této situace je jasný: pro přesnější vyhledání globálního minima je vhodné zvolit takovou hodnotu delta, jejíž počet desetinných míst bude stejný nebo menší než počet desetinných míst náhodně generovaného x. Zároveň, v případě volby delta = 0,01, by mělo být zcela dostačující nastavit proměnnou itermax na hodnotu větší než 3. Počet k – v algoritmu jako proměnná citac lze vzhledem k intervalu a deltě vyjádřit matematicky, a to jako interval v jednotkách * 1/delta, tedy konkrétně:

(5,12 – (-5,12)) * 1 / 0,01 = 1024

Toto omezení koresponduje s údaji v tabulkách 4 a 6, kde jsou všechny její hodnoty k menší než námi vypočítaná.

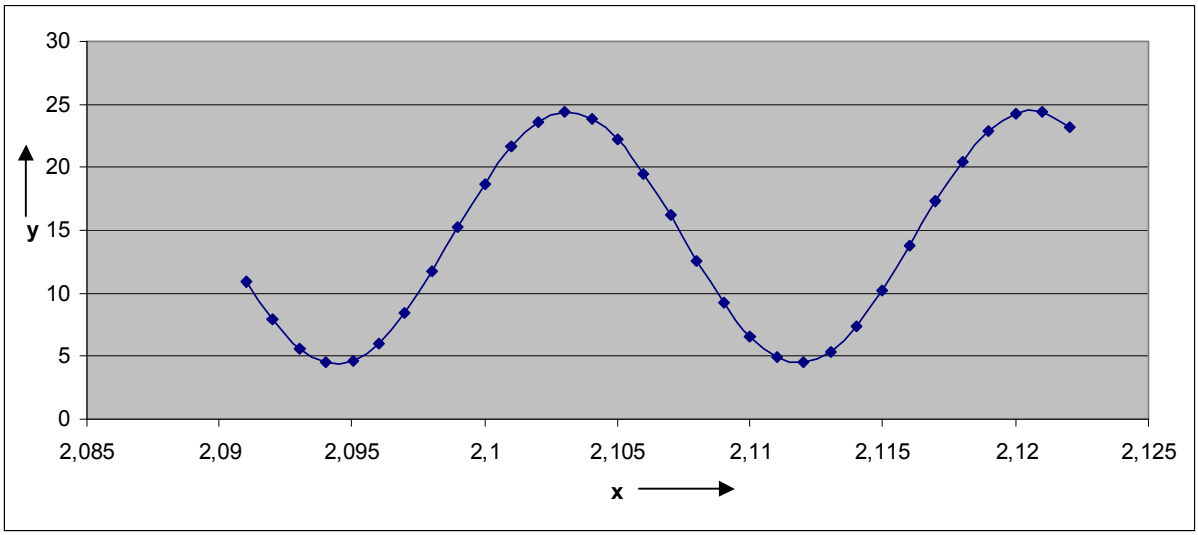
14.1 Velikost Tabu seznamu v 1 rozměrném poli

V průběhu šetření jsme se nezaměřili na délku Tabu seznamu. Důvod je zřejmý – u jednorozměrného prostoru si můžeme funkci představit jako vektor, ve kterém lze jít pouze dvěma směry. Vzhledem k tomu, že se při druhém průchodu zaznamená hodnota do Tabu seznamu, a tudíž tím směrem již nelze jít, nastavení délky zakázaného seznamu na velikost 1 je dostačující, a to pro všechny funkce.

U dvourozměrného prostoru již tato velikost není jednoznačná a v další kapitole bude podrobena šetření.

Jestliže deltu určíme 0,001, budeme pracovat obdobným způsobem. Tuto hodnotu použijeme pro případ, že x má 3 desetinná místa. Pokud by měla pouze dvě, jak bylo určeno v předchozí kapitole, dobrali bychom se stejných výsledků, pouze s rozdílem času a počtu iterací celkových a akceptovatelných. Při zachování hodnoty itermax by zcela jistě ve fázi úniku z lokálního minima došlo k terminaci, neboť by byla splněna podmínka iter > itermax (nastavená na 3).

Podle vyobrazeného grafu na obrázku 18 a výsledků šetření v tabulce 7 můžeme opět určit optimální hodnotu itermax, stejně tak jako hodnotu k, kterou můžeme opět vypočítat pomocí vzorce.



Obr. 18 Graf Rastriginovy funkce pro delta = 0,001 v okolí bodu 2,105

Tab. 7 Výsledky algoritmu TS na Rastriginově funkci pro delta = 0,001

xpocatecni	k	xbest	ybest	cas	itermax
-0,753	7882	0,000	0,000	0,03125	9
-0,243	5668	0,000	0,000	0,015625	9
-3,486	1637	-3,492	12,262	0,015625	9
-3,212	8337	0,000	0,002	0,03125	9
4,770	9895	0,087	0,014	0,03125	9
-1,383	9300	0,000	0,000	0,03125	9
2,999	2124	3,002	9,013	0	9
3,652	8777	0,000	0,004	0,046875	9
-3,954	1169	-3,962	15,704	0	9
-1,477	3646	-1,501	2,253	0,015625	9
3,418	9709	0,000	0,003	0,046875	9
-4,561	9686	0,000	0,014	0,03125	9
2,987	8112	0,000	0,001	0,03125	9
-1,057	4066	-1,082	1,178	0,015625	9
1,454	7367	0,000	0,000	0,03125	9
-0,717	5842	0,001	0,000	0,015625	9
4,420	9545	0,000	0,012	0,03125	9
0,861	7990	-0,001	0,000	0,03125	9
3,658	1465	3,665	13,456	0	9
3,026	8151	-0,001	0,001	0,03125	9

Tab. 8 Počet výsledků přibližujících se globálnímu minimu

minimum	absolutní počet	relativní počet
0	29	56,86%
>0	22	43,14%
celkem	51	100,00%

V tabulce 8 je přehled výsledků z šetření. Ve 29 případech z 51 se algoritmus dobral výsledku globálního minima. Stejně tak bylo provedeno šetření i pro deltu 0,0001 a 0,00001. Výsledky jsou zaznamenány v tabulce 9.

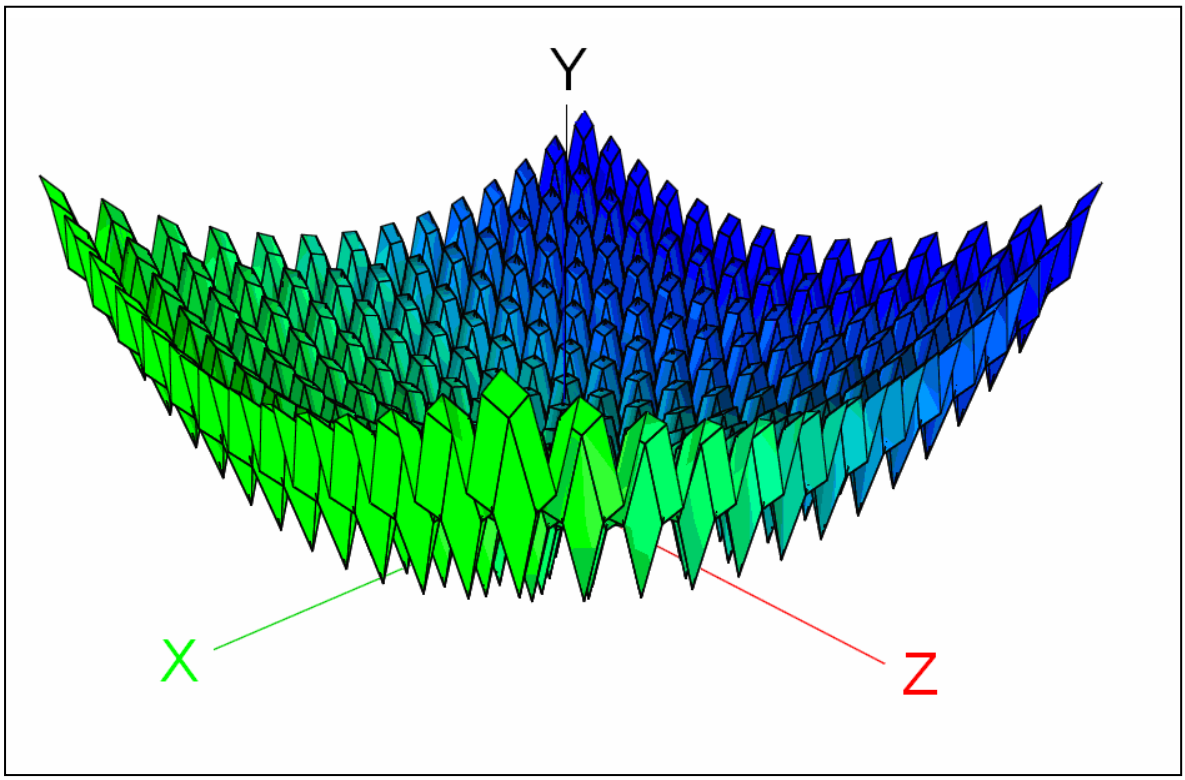
Tab. 9 Vliv delta na počet iterací

delta	k	itermax
0,01	1024	2
0,001	10240	9
0,0001	102400	88
0,00001	1024000	874

Závěr z těchto měření je následující:

- počet k, omezení cyklů, lze spočítat pomocí vzorce k = délka intervalu * (1/delta)
- podmínku pro počet akceptovatelných tahů, jejichž funkční hodnota je vyšší než aktuální, lze vyjádřit vzorcem itermax = 0,02/ delta
- délka Tabu seznamu i(T) = 1

15 Rastriginova funkce v dvourozměrném prostoru



Obr. 19 Rastriginova funkce v dvourozměrném prostoru

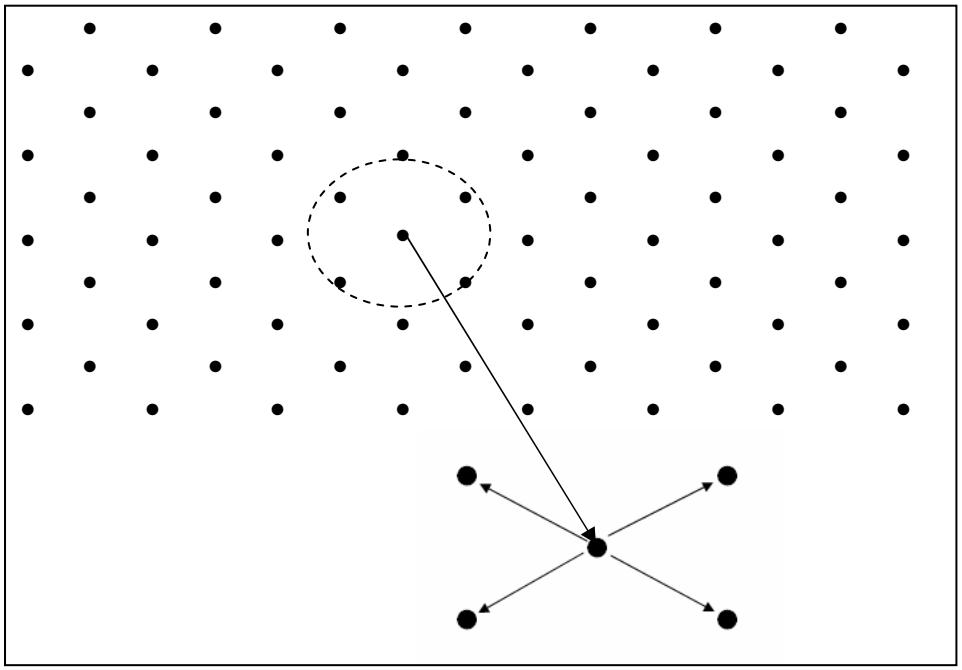
Matematický zápis funkce na obrázku 19 vypadá následovně:
20 + ((x * x) - 10 * (Cos(360 * x)) + ((y * y) - 10 * (Cos(360 * y))))
Globální minimum s funkční hodnotou 0 je v bodě [0;0].

15.1 Parametry dvourozměrné funkce

Tato kapitola je zaměřena především na určení optimální velikosti Tabu seznamu. Další proměnné získané v předchozí kapitole v jednorozměrném prostoru postačí matematicky upravit a to:

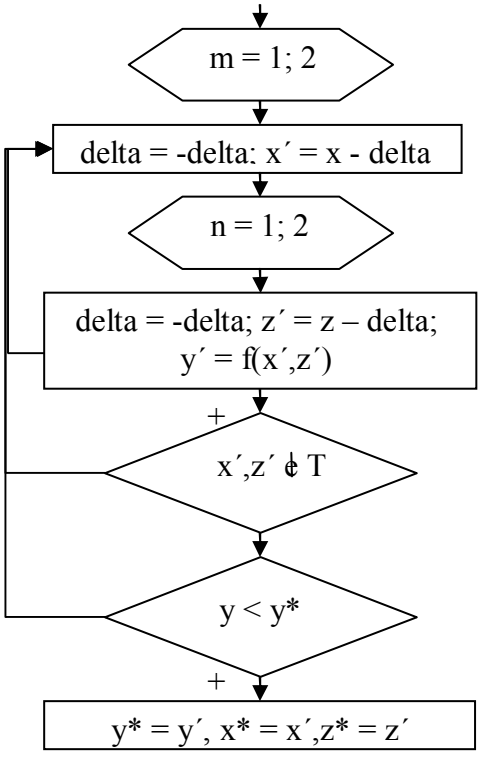
- počet k = ((délka intervalu x) * (1/delta)) * ((délka intervalu z) * (1/delta))
- podmínku pro počet akceptovatelných tahů, jejichž funkční hodnota je vyšší než aktuální, lze vyjádřit vzorcem itermax = 0,01/ delta

Pro lepší představu je část funkce s jejími prvky znázorněna pohledem shora. Na výřezu z obrázku 19 je patrný výběr kandidátů z aktuálního prvku.



Obr. 20 Prvky dvourozměrné funkce

Pro tento výběr je nutné upravit algoritmus z předchozí kapitoly použitý pro prohledávání v jednorozměrném prostoru. Do tohoto algoritmu je přidán další For cyklus, ve kterém bude invertována hodnota delta pro osu z, tedy stejným způsobem jako pro osu x, a následně porovnávána hodnota účelové funkce y´ ze získaného tahu [x´,z´] s aktuálním. Takto upravený algoritmus je popsán na obrázku 21.



Obr. 21 Algoritmus TS dvourozměrné funkce

15.2 Velikost Tabu seznamu

V případě volby hodnoty delta 0,1 by měla být velikost zakázaného seznamu pro 40 prvků dostačující. Na začátku testování byla tato hodnota naddimenzována, a pokud při testování nedocházelo k zacyklení, byla hodnota snižována.

15.3 Výsledky šetření Rastriginovy dvourozměrné funkce

Podle matematického výpočtu velikosti intervalu, kdy počet všech prvků = (interval / delta)^2, tedy v tomto případě 10404 prvků, byla hodnota iterací nastavena na dostačující velikost 1000. Hodnoty, ze kterých se vycházelo, jsou uvedeny v tabulce 10.

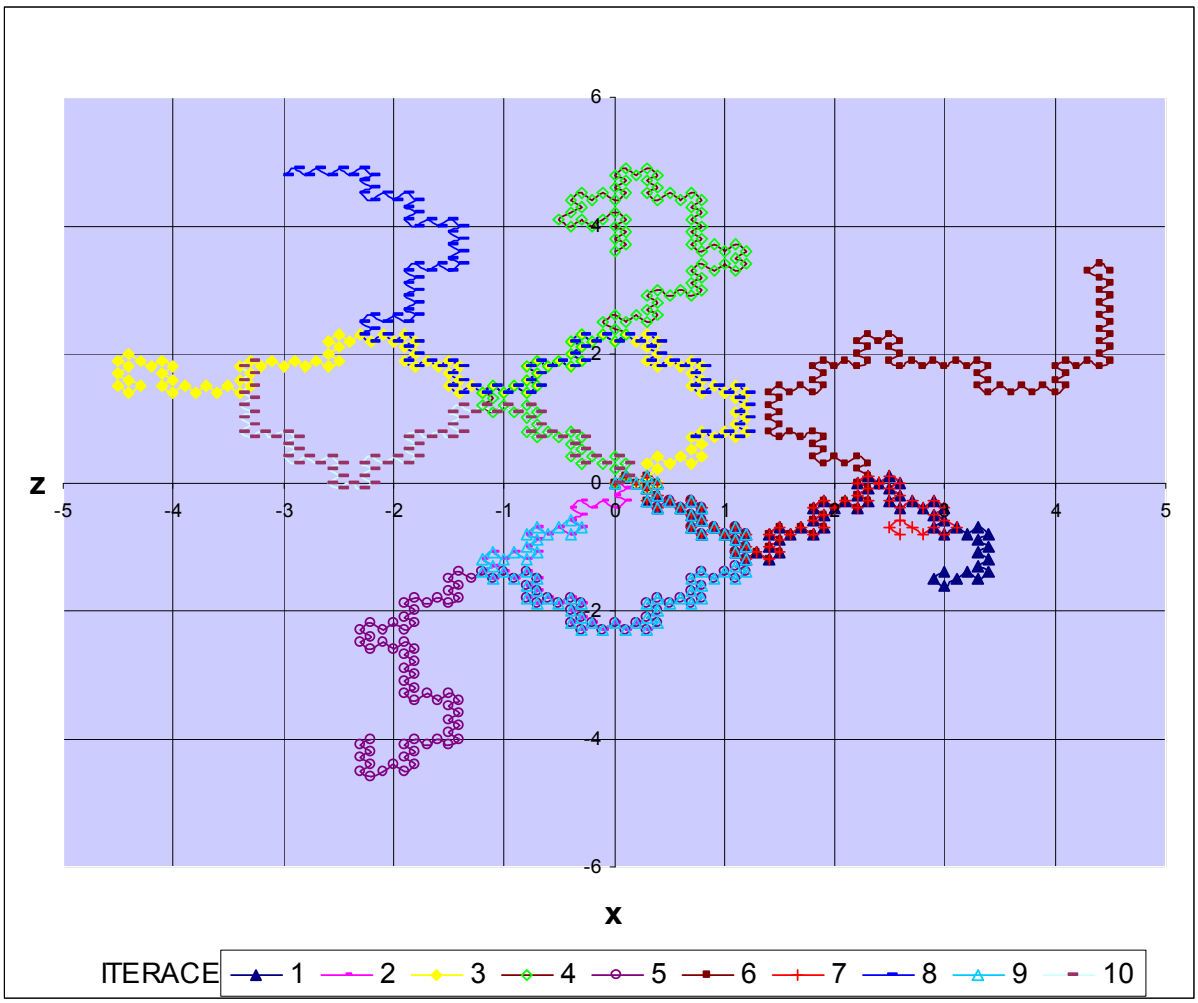
Tab. 10 Počáteční hodnoty Rastriginovy funkce

parametr	hodnota
delta	0,1
interval	<-5,1;5,1>
velikost intervalu	10,2
počet prvků celého intervalu	10404
velikost tabu seznamu	40

Bylo provedeno celkem 50 pokusů, z nichž výsledky prvních deseti jsou uvedeny v tabulce 11, na obrázku 22 je vyobrazena trajektorie prohledávání.

Tab. 11 Výsledky šetření Rastriginovy dvourozměrné funkce

k	xbest	zbest	ybest	cas	itermax	xpocatecni	zpocatecni
81	0	0	0	0,015625	3	2,9	-1,3
52	0	0	0	0,015625	3	0,4	-2,2
128	0	0	0	0,03125	4	-4,6	1,6
121	0	0	0	0,015625	3	-0,1	3,7
131	0	0	0	0,015625	3	-2,1	-4,1
138	0	0	0	0,03125	3	4,6	3,4
71	0	0	0	0,015625	3	2,5	-0,9
137	0	0	0	0,03125	3	-2,9	4,7
85	0	0	0	0,015625	3	-0,5	-0,9
80	0	0	0	0,015625	3	-3,2	1,8



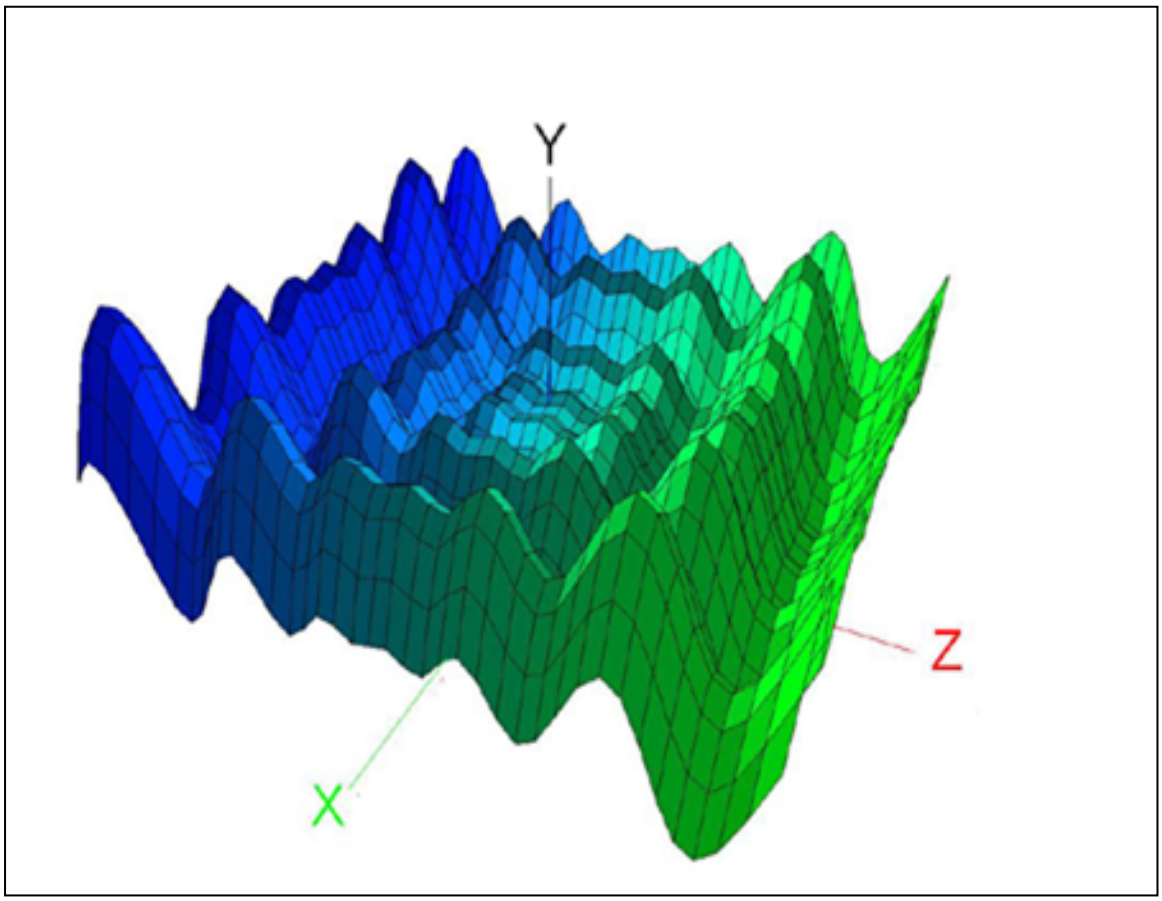
Obr. 22 Trajektorie prohledávání dvourozměrné Rastriginovy funkce

proměnná	max	min	průměr
k	180	42	100,52
cas	0,03125	0	0,01625
itermax	4	3	3,04
xbest	0	0	0
zbest	0	0	0

Tab. 12 Statistika výsledků Rastriginovy dvourozměrné funkce

Výsledky z 50 pokusů jsou uvedeny v tabulce 12. Z tabulky statistiky je zřejmé, že počet průchodů byl nastaven dostatečně. Z průměrné hodnoty itermax lze odvodit velikost zakázaného seznamu jako (itermax x 2)^2, tedy (3*2)^2 = 36. Velikost zakázaného seznamu byla také dostačující.

16 Schwefelova funkce v dvourozměrném prostoru



Obr. 23 Schwefelova funkce v dvourozměrném prostoru

$$f(x) = 418{,}9820n - \sum_{n=1}^{n} (x_i \sin\sqrt{|x_i|})$$

Průběh této funkce je vidět z obrázku 23. Interval, ve kterém bude funkce prověřována, bude nastaven od -500 do 500. Matematický zápis této funkce vypadá následovně:

(418.9829*2)-((x*sin(sqrt(abs(x)))) +(z*sin(sqrt(abs(z)))))

Globální minimum s funkční hodnotou 0 je v bodě [420,9687; 420,9687]. V tabulce 13 jsou uvedeny parametry, které budou použity pro testování.

Tab. 13 Parametry Schwefelovy funkce

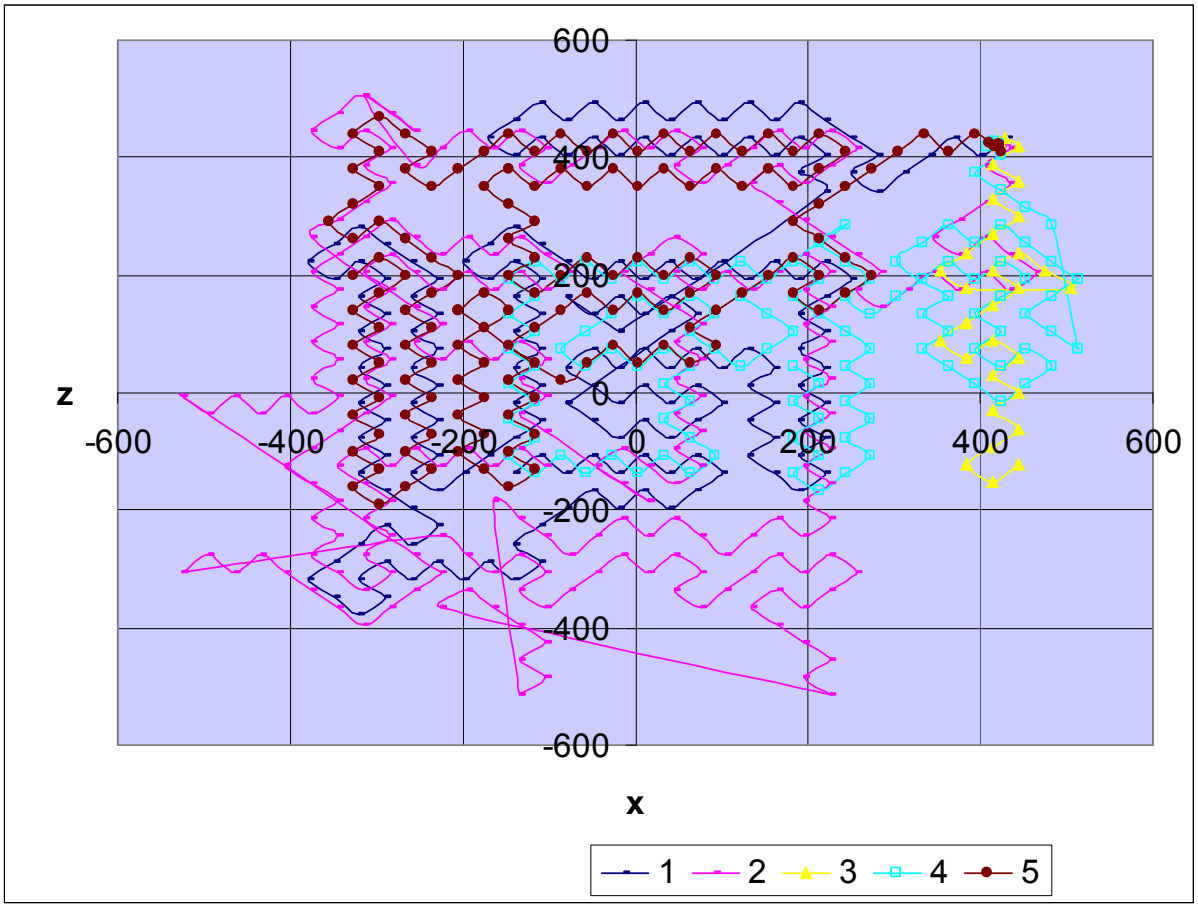
parametr	hodnota
delta	30
interval	<-500;500>
velikost intervalu	1000
počet prvků celého intervalu	1111,111111
velikost tabu seznamu	300

V průběhu testování nastal zásadní problém, který je zřejmý při porovnání hodnoty delta a globálního minima. Výsledky nalezeného minima se pouze přibližovaly skutečnému minimu. Z několika prvních pokusů byl určen práh funkční hodnoty, kdy se při jeho dosažení velikost delty geometricky snižovala. Tím bylo zajištěno detailnější prohledání nadějného minima.

Opačný princip, tedy zvyšování hodnoty delta, bylo třeba zavést z důvodů zacyklení, tedy přesněji v situaci, kdy se aktuálním bodem stal prvek, jehož sousedi byli v Tabu seznamu. Po úniku z této pasti se hodnota delta vrátila na svoji původní hodnotu. Výsledky šetření prvních deseti pokusů jsou uvedeny v tabulce 14. Pro přehlednost bylo do obrázku 24 trajektorie zahrnuto pouze prvních pět pokusů.

Tab. 14 Výsledky šetření Rastriginovy dvourozměrné funkce

k	xbest	zbest	ybest	cas	itermax	xpocatecni	zpocatecni
201	420,9618	420,9736	0	0,359375	5	-110	-167
265	420,9741	420,9691	0	0,484375	7	-135	-155
40	420,958	420,9774	0	0,171875	4	474	-151
133	420,9786	420,9628	0	0,25	4	213	74
157	420,9678	420,9718	0	0,296875	5	-356	230
495	420,9824	420,9668	0	0,859375	6	49	435
471	420,9668	420,9786	0	0,875	10	-345	48
81	420,9649	420,9649	0	0,15625	3	-308	-23
239	420,9736	420,9696	0	0,421875	6	313	-3
88	420,9696	420,9718	0	0,15625	4	222	425



Obr. 24 Trajektorie prohledávání dvourozměrné Schwefelovy funkce

Výsledky z 50ti pokusů jsou uvedeny v tabulce 15.

Tab. 15 Statistika výsledků Rastriginovy dvourozměrné funkce

proměnná	max	min	průměr
k	495	16	169,9
cas	0,875	0,046875	0,305313
itermax	10	1	4,58
xbest	420,9824	420,9562	420,9701
zbest	420,9805	420,9595	420,9698

Závěr

Cílem této práce bylo nastínit problematiku optimalizačního algoritmu TS, zpracovat a sestavit algoritmus v programovém prostředí Microsoft Access, v neposlední řadě také přesvědčit, že je funkční.

Pro demonstraci implementace metody zakázaného prohledávání bylo zvoleno hledání globálního minima pro jednorozměrnou a dvourozměrnou Rastriginovu a Schwefelovu funkci. Bylo zjištěno, že tato metoda je schopna daný problém řešit, avšak bylo zapotřebí experimentovat s parametry.

Velmi důležitým parametrem je délka zakázaného seznamu, neboť tato hodnota ovlivňuje schopnost vymanit se z lokálního extrému, v této práci tedy minima. Pokud byl parametr nastaven na hodnotu nižší než optimální, docházelo při šetření k zacyklení (přiblížení se výsledkům algoritmu Hill Climbing), u příliš vysokého se pak snižovala rychlost výpočtu.

Na začátku práce bylo třeba vyřešit problém, jakým nejefektivnějším způsobem realizovat zakázaný seznam. Jedna z možností byla zapisovat a modifikovat získaná data do tabulky pomocí objektu Recordset. Tento způsob se ovšem z důvodů rychlosti výpočtu neosvědčil jako nejlepší, proto byl použit datový typ fronta. Tento typ má v prostředí Microsoft Access pouze jedno omezení, neboť nedokáže v průběhu výpočtu měnit svou velikost, jelikož se při opětovné deklaraci, tedy změny velikosti seznamu, ztratí veškerá data. Toto omezení tedy bylo akceptováno a v průběhu testování byl parametr délky Tabu seznamu měněn ručně. Vzhledem k tomu, že metoda je založena na opakovaném spuštění prohledávání, lze velikost měnit mezi tímto opakováním.

Velikost parametru delta (rozdílová hodnota mezi tahy), silně ovlivňuje konečný výsledek řešení. V případě volby vysoké hodnoty nejlepší výsledek zpravidla inklinoval ke skutečně nejlepší hodnotě. Při malé hodnotě byl výpočet příliš náročný, resp. docházelo k průchodu velkého množství prvků. Při testovacích výpočtech s vysokou hodnotou delty byl nastaven práh účelové funkce. Ve chvíli, kdy se hodnota účelové funkce dostala pod tento práh, byla snížena velikost parametru řádově na desetinu. Tím bylo dosaženo poměrně přesnějších výsledků.

Vzhledem ke způsobu výběru kandidátů docházelo při nízkých hodnotách delty k zacyklení, a to z důvodu, že všichni okolní aspiranti byli v Tabu seznamu. Pro tento případ

byla zvyšována hodnota delta, aby došlo k přesunu do prostoru, kde alespoň jeden prvek v Tabu seznamu nebyl.

Nevýhodou deterministického přístupu, kdy algoritmus akceptuje lepší řešení než stávající, je v případě jednorozměrné funkce, protože může nalézt pouze nevýrazné minimum lokální. To je způsobeno výběrem lepšího prvku od počátečního náhodného řešení, kdy se algoritmus vydá určitým směrem. Pojem nevýhoda je ovšem vágní, neboť při testování funkce, která nemá charakter konvexnosti, je určený směr odvozený z lokálního minima vyhledávání zcela v pořádku. Pro testovanou funkci lze opětovným spuštěním algoritmu docílit slušných výsledků v nalezení minima globálního, a tím tento nedostatek odstranit.

Data získaná v průběhu šetření sloužila jako podklad pro ověření matematických výpočtů pro určení vhodné velikosti parametrů při implementaci. Po výpočtu těchto hodnot vycházejících z testované funkce bylo provedeno spuštění algoritmu v řádech několika desítek a výsledky byly porovnány s výpočty, které zcela korespondovaly, respektive hodnoty výsledků se pohybovaly v intervalech výpočtů.

Ve své praxi se setkávám s tvorbou databází v prostředí Microsoft Access, které je určeno spíše pro práci s tabulkami a formuláři. Zajímalo mě, zda je možno tuto metodu implementovat do tohoto prostředí. V budoucnu bych určitě velmi rád tuto metodu vyzkoušel na příkladu z praxe, neboť vytvořený algoritmus by byl s modifikovanými parametry plně schopen daný problém vyřešit. Omezení by byla ohraničena matematickými funkcemi, ve kterých by se optimalizace musela držet a určení velikosti zakázaného seznamu by bylo možné docílit testováním na vzorku dat.

Tato práce nám ukázala, že algoritmus TS je v prostředí Microsoft Access zcela plnohodnotným, kvalitním a využitelným algoritmem.

Použitá literatura

[1] GLOVER, Fred. Future Paths for Integer Programming and Links to Artificial Intelligence, Computer and Operations Research [online]. 1986 [cit. 2009-02-12]. Dostupný z WWW: <http://leeds-faculty.colorado.edu/glover/TS - Future Paths for Integer Programming.pdf>

[2] GLOVER, Fred, LAGUNA, Manuel. Tabu Search. 6 aktualiz. vyd. Dodrecht: Kluwer Academic Publisher 1997, 408 s. ISBN 0-7923-8187-4

[3] BARTÁK, Roman. Programování s omezujícími podmínkami [online]. 2008 [cit. 2009-02-16]. Dostupný z WWW: <http://kti.mff.cuni.cz/~bartak/podminky/lectures/lecture02.pdf>

[4] GENDREAU, Michel. An introduction to Tabu search [online]. 2002 [cit. 2009-02-16]. Dostupný z WWW: <http://www.ifi.uio.no/infheur/Bakgrunn/Intro_to_TS_Gendreau.htm>

[5] GLOVER, Fred. Tabu Search - Part I [online], ORSA Journal on Computing 1989 [cit. 2009-02-22]. Dostupný z WWW: < leeds-faculty.colorado.edu/glover/TS – Part I-ORSA.pdf>

[6] ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. Praha: BEN-technická literatura, 235 s. ISBN 80-7300-069-5

[7] MICHIELS, Wil, AARTS, Emile, KORST, Jan. Theoretical Aspects of Local Search, Springer -Verlag New York 2007, 283 s. ISBN: 978-3-540-35853-4

[8] GLOVER, Fred. Tabu Search - Part II [online], ORSA Journal on Computing 1989 [cit. 2009-03-10]. Dostupný z WWW: <leeds-faculty.colorado.edu/glover/TS – Part II-ORSA.pdf>

[9] OPPEN, Johan, GRUNER, Solveig Irene, LOKKETANGEN, Arne. A tabu search based heuristic for the 0/1 Multiconstrained [online]. 2005 [cit. 2009-03-19]. Dostupný z WWW: <http://www.nik.no/2003/Bidrag/Oppen.pdf>

[10] HERTZ, Alan, TAILLARD, Eric, WERRA, De Dominique. A tutorial on Tabu search [online]. [cit. 2009-03-20]. Dostupný z WWW: <http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf>

[11] SCHMIDT. Tabu prohledávání [online]. [cit. 2009-03-20]. Dostupný z WWW: <makovice.nipax.cz/FEL/PAA/prednasky/PAA11tabu.pdf>

[12] MISIVIČEUS, Alfonsas. Using iterated Tabu search for the traveling salesman problem 2004 [cit. 2009-03-24]. Dostupný z WWW: <http://itc.ktu.lt/itc32/Misev32.pdf>

[13] KUNZ, Kerstin. A comparison of heuristic search algorithms for molecular docking [online] 1997 [cit. 2009-03-27]. Dostupný z WWW: <fred.bioinf.uni-sb.de:4711/proseminar_ss05/vortraege/KKunz_Presentation.ppt>

Seznam obrázků

Seznam tabulek