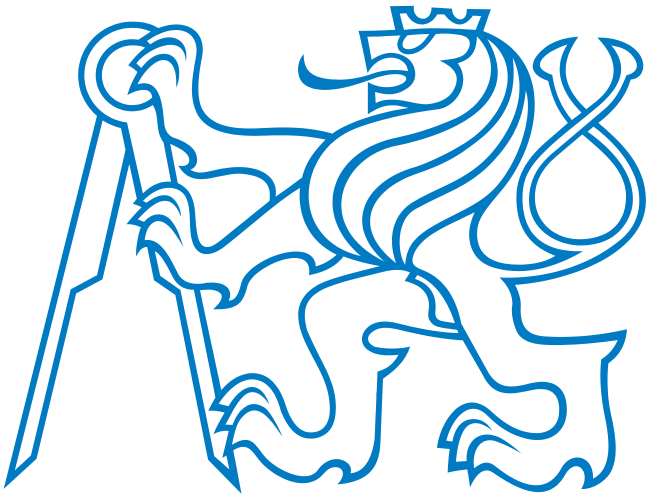
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

BAKALÁŘSKÁ PRÁCE



Přemysl Kafka

Samoorganizující se neuronové sítě v úlohách plánování cesty přes více cílů

Katedra kybernetiky

Vedoucí bakalářské práce: **Ing. Jan Faigl, Ph.D.**

Praha, 2011

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, SW, projekty atd.) uvedené v přiloženém seznamu.

V Praze dne . . . . . .

. . . . . .
podpis

Abstrakt

Bakalářská práce se zabývá studiem vlivu grafu reprezentujícího polygonální doménu na kvalitu řešení úlohy obchodního cestujícího řešené algoritmem samoorganizující se sítě na grafu. Jako výchozí grafové reprezentace byly zvoleny tři základní typy grafů lišící se způsobem své konstrukce: (1) graf trojúhelníkové mřížky, (2) graf viditelnosti a (3) graf rostoucího neuronového plynu (GNG). Posledně jmenovaný graf je studován v práci podrobněji s cílem nalézt co nejvhodnější parametry algoritmu neuronového plynu tak, aby byl výsledný graf GNG co možná nejvhodnější pro samoorganizující se síť řešící problém obchodního cestujícího, tj. aby vedl na kvalitní řešení nalezené v co možná nejkratším čase. Použití GNG v polygonální doméně vyžaduje respektování překážek, neboť přípustné řešení úlohy obchodního cestujícího musí ležet ve volném prostoru. Proto bylo navrženo několik rozšíření algoritmu GNG, které byly experimentálně ověřeny na vybraných problémech. V závěru práce je uvedeno porovnání vlivu různých grafů na kvalitu řešení úlohy obchodního cestujícího a zhodnocení přínosů grafové reprezentace.

Abstract

The thesis deals with an effect of a graph representing the polygonal domain to the solution quality of the traveling salesman problem that is solved by an algorithm of self-organizing map on a graph. Three types of graphs, which are created by different approaches, were selected: (1) Triangular-mesh graph; (2) Visibility graph, (3) and Growing neural gas graph (GNG). The last mentioned graph is studied in a more detail as one of the main goal of this bachelor thesis is to find parameters of the growing neural gas algorithm in order to create a graph suitable for the self-organizing map based algorithm for the traveling salesman problem. Although the GNG approach is well known, its application for describing the polygonal domain is not straightforward because of obstacles. A feasible solution of the traveling salesman problem has to be within the polygonal domain representing free space for a mobile robot. Therefore, several approaches to consider obstacles in the GNG algorithm have been proposed and experimentally examined in a set of selected problems. A comparison of various graphs influence on the solution quality of the traveling salesman problem is presented in the thesis. Benefits of the graph representation to the solution quality and computational requirements are discussed in the conclusion.

Můj velký dík patří vedoucímu této bakalářské práce panu Ing. Janu Faiglovi, Ph.D., který se velkou měrou zapříčinil o její vznik a zvláště pak bych mu chtěl poděkoval za nenahraditelnou pomoc v závěrečné fázi bakalářské práce. Dále bych chtěl poděkovat všem těm, co mi jakkoliv pomohli při vytváření této práce a všem, co mi vyjádřili potřebnou podporu.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Problém obchodního cestujícího (TSP) je klasický problém, jehož cílem je nalézt vhodnou cestu mezi zadanými body (městy). Tento problém je řazen mezi takzvané NP úplné úlohy. Jedná se o úlohy, jejichž výsledky lze ověřit v polynomálním čase, ale není dokázáno, zda pro ně lze nalézt i řešení v polynomálním čase. Je zřejmé, že problém obchodního cestujícího je časově náročný, avšak má důležité uplatnění v praxi, a proto je potřeba tento problém studovat.

Algoritmy pro řešení problému obchodního cestujícího mohou být rozděleny do dvou základních skupin. A to na deterministické algoritmy a heuristické algoritmy. Deterministické algoritmy vždy naleznou nejlepší možné řešení, bohužel jsou však výpočetně velmi náročné. To se projevuje u rozsáhlejších problémů (větší množství měst určených k navštívení), proto v těchto případech nemusí být použití deterministických algoritmů vhodné. Naproti tomu heuristické algoritmy jsou poměrně rychlé, ale nezaručují nalezení optimálního řešení. Avšak jejich použití v praxi může být výhodné, protože s nimi lze nalézt dostatečně kvalitní řešení v krátkém čase. Takovéto algoritmy se využívají například v mobilní robotice, kde není preferováno optimální řešení za každou cenu, ale naopak je kladen důraz na to, aby se robot rozhodoval co nejrychleji a s přijatelnou kvalitou řešení.

Kromě klasických kombinatorických přístupů jsou studovány takzvané softcomputingové techniky, mezi které patří genetické algoritmy, mravenčí kolonie a také neuronové sítě. Právě neuronovými sítěmi, konkrétně samoorganizujícími se mapami (SOM), se zabývá tato bakalářská práce. Přestože je známo, že SOM přístupy pro úlohu obchodního cestujícího jsou obecně pomalejší a poskytují horší výsledky než sofistifikované heuristiky, existují případy, kdy je jejich použití výhodné. Například pokud cíle obchodního cestujícího nejsou body, ale třeba úsečky [4] či polygony [5]. Využití těchto útvarů, jakožto cílů, je vhodné například při prohledávání prostředí tak, aby bylo celé prozkoumáno [2].

Prostředí, v němž hledá obchodní cestující cestu, nemusí být vždy bez překážek. V reálném světě jsou překážky časté, proto je nutné algoritmus SOM tomuto problému přizpůsobit. V tomto textu je využito principu popsání volného prostředí polygonální doménou, přesněji nějakým grafem. Tento graf determinuje obchodního cestujícího a je tak zajištěno, že se nebude nikdy vyskytovat v prostoru překážky.

Druhů grafů, které mohou reprezentovat volný prostor, je mnoho. Každý graf má odlišné vlastnosti a pro řešení problému obchodního cestujícího se hodí více či méně. Tato bakalářská práce se proto zabývá vlivem grafu na výsledek problému obchodního cestujícího. Jako grafová reprezentace polygonální domény reprezentující prostředí byly zvoleny následující grafy: graf viditelnosti, graf trojúhelníkové mřížky a graf rostoucího neuronového plynu (GNG). Posledním zmiňovaným grafem se text zabývá podrobněji a snaží se určit jeho nejvhodnější parametry grafu, jakožto vstupu algoritmu SOM.

Bakalářská práce je rozdělena do tří kapitol. V první kapitole je pojednáváno o samot-

ném problému obchodního cestujícího a jeho modifikacích či řešení problému algoritmem samoorganizující se sítě. Další kapitola se zabývá neuronovými sítěmi, jakožto reprezentací polygonální domény pro problém obchodního cestujícího. Poslední kapitola obsahuje výsledky experimentů a jejich zhodnocení.

Kapitola 1

Problém obchodního cestujícího

1.1 Definice problému obchodního cestujícího

Nechť je dána konečná množina vrcholů $\mathbf{V} = \{v_1, v_2, ..., v_n\}$ a vrchol $v_d \in \mathbf{V}$ se jmenuje depot. Množina vrcholů $\mathbf{V}$ určuje vrcholy grafu $G = (V, E, \varepsilon)$, kde $\mathbf{E}$ je konečná množina hran mezi vrcholy (městy) grafu G a ε je vztah incidence. Pokud je graf G orientovaný, je ε přiřazení, které každé hraně $e \in E$ přiřazuje uspořádanou dvojici vrcholů. Je-li graf G neorientovaný, poté je ε přiřazení, které každé hraně $e \in E$ přiřazuje množinu $\{u, v\}$ pro vrcholy $u, v \in \mathbf{V}$. A každá hrana $e_{u,v}$, kde $e_{u,v} \in E$ má cenové ohodnocení $c_{u,v}$. Úkolem úlohy je nalézt nejkratší cestu (určenou posloupností vrcholů) začínající a zároveň končící vrcholem v_d, jež musí obsahovat každý vrchol z množiny $V \setminus \{v_d\}$ právě jednou.

Existuje-li přímá cesta z bodu A do bodu B a neexistuje-li kratší cesta z bodu A do bodu B, která by zároveň vedla i přes bod C, jež se nenachází na přímé cestě z A do B, poté v grafu platí trojúhelníková nerovnost. V takovém případě se jedná o metrický problém obchodního cestujícího a ten může být dále dělen:

1. Ohodnocení hran je udáno Euklidovskou vzdáleností, což znamená, že vzdálenost mezi městem a a městem b je dána vztahem

$$\| a, b \| = \sum_{i=1}^{n} \sqrt{(a_i - b_i)^2}, \tag{1.1}$$

   kde i vyjadřuje osu souřadnice. Takovéto ohodnocení hrany představuje nejkratší možnou vzdálenost mezi dvěma body.

2. Ohodnocení hran je dáno tzv. Manhattanskou vzdáleností (nebo také New Yorskou vzdáleností), pro kterou platí, že vzdálenost mezi vrcholem a a vrcholem b je definována

$$\| a, b \| = \sum_{i=1}^{n} |(a_i - b_i)|, \tag{1.2}$$

   kde i vyjadřuje osu souřadnice. Manhattanská vzdálenost se například využívá u strojů, které se pohybují v daný okamžik pouze po jedné ose. Jako ilustrace může sloužit jeřáb, pohybující se po blocích skladiště.

3. Ohodnocení hran je vyjádřeno Čebyševovou vzdáleností

$$\|a, b\| = \max_k |a_k, b_k|, \tag{1.3}$$

která udává maximum z rozdílů souřadnic pro jednotlivé osy. Tato metrika je využívána například u stojů, které nastavují své souřadnice současně, proto čas spotřebovaný pro pohyb je dán maximální dobou posunu po některé z os. Jako příklad může být uvedena automatizovaná vrtačka.

Pokud je ohodnocení hrany určeno vzdáleností mezi vrcholy a zároveň i jiným parametrem (např. autonavigace upřednostňující delší cestu po dálnici před kratší cestou po okresních silnicích), nebo ohodnocení hran není vůbec udáváno vzdáleností mezi vrcholy, ale jinou charakteristikou, která spojuje vrcholy, neplatí v grafu trojúhelníková nerovnost a tento problém se označuje jako ne-metrický problém obchodního cestujícího.

Dále může být TSP dělen na symetrický problém obchodního cestujícího a asymetrický problém obchodního cestujícího. Vstupem algoritmu pro symetrický problém obchodního cestujícího je neorientovaný graf, který reprezentuje cesty mezi městy. V tomto textu bude myšlen symetrický problém obchodního cestujícího, nebude-li řečeno jinak.

Pro asymetrický problém obchodního cestujícího je vstupem algoritmu orientovaný graf. Jako příklad asymetrického grafu může být považována reprezentace cesty, která vede přes kopec. Pojede-li obchodní cestující směrem do kopce, bude mít v tomto směru hrana větší cenové ohodnocení než v opačném směru a naopak.

V této práci je TSP modifikován pro pohyb obchodního cestujícího v polygonální doméně. Polygonální doména je zde reprezentována grafem $G_d = (V_d, E_d, \epsilon_d)$ pro který platí, $\mathbf{V} \subseteq \mathbf{V_d}$, kde $\mathbf{V}$ je množina měst z původního TSP. Z tohoto důvodu jsou všechny vzdálenosti a cesty mezi městy TSP určeny na grafu G_d.

1.2 Využití problému obchodního cestujícího v praxi

1.2.1 Aplikace v praxi

TSP má asi nejznámější uplatnění v logistice, například problém nalezení nejlepší trasy pro školní autobus, který má za úkol vyzvednout děti v určité čtvrti a dopravit je do školy (tato aplikace je historicky důležitá pro problém obchodního cestujícího, protože motivovala Merrilla Flooda k prvnímu širšímu zkoumání TSP (rok 1940)).

Dalším využitím v praxi bylo nalezení nejlepší cesty, kterou měly absolvovat zemědělské stroje tak, aby co nejefektivněji obdělaly půdu. To vedlo k dalším matematickým studiím P.C. Mahalanobise či R.J. Jessena. V současnosti se řešení úlohy používá například pro nalezení optimální trasy opravářů kabelů kabelových firem, či pro dovážení jídla domů lidem, kteří s obtížemi opouštějí domov. Dále se využívá při plánování trasy jeřábů ve skladech, nebo směřování vozíků zajišťujících balíkovou poštu.

Z čistě technických příkladů využití TSP je možné uvést analýzu struktury krystalů, opravu turbíny plynových motorů, či slučování dat z polí. Z uvedených aplikací je patrné významné využití TSP v praxi, z čehož plyne, že jeho studium má velký význam.

1.2.2 Využití TSP při konstrukci evolučního stromu

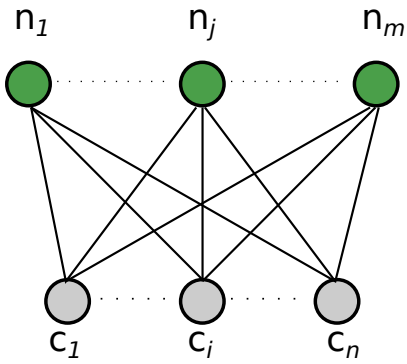
TSP je uplatňován v různorodých oborech. Jedním z nich je i biologie, která zkoumá vztahy mezi živočichy a pro studium těchto vztahů je důležitá znalost evolučního stromu.

Evoluční strom pochází z Darwinovy teorie evoluce a zobrazuje vztahy mezi jednotlivými biologickými druhy, které mají podobné fyzické nebo genetické vlastnosti a předpokládá se u nich společný předek. Tímto způsobem se nejen zkoumá evoluční vývoj biologický druhů, ale například i genetický vývoj virů AIDS v jednotlivých stádiích onemocnění. Tyto informace dále slouží k vývoji vakcíny proti této nemoci.

Samotná metoda konstrukce evolučního stromu hledá takový strom, jehož posloupnost vrcholů, které jsou reprezentovány posloupností proteinů, má minimální cenu. Ohodnocení posloupností proteinů je určeno takzvanou PAM vzdáleností[1]. Řešení konstrukce evolučního stromu je zredukováno na použití algoritmu obchodního cestujícího. Vstupem do algoritmu obchodního cestujícího jsou zmíněné PAM vzdálenosti mezi sekvencemi proteinů a výstupem je optimální cesta mezi sekvencemi proteinů a ohodnocení této cesty. Získaná cesta a ohodnocení je následně použita pro samotné zkonstruování optimálního evolučního stromu.

1.3 Samoorganizující se síť pro problém obchodního cestujícího

Hlavním přístupem pro řešení TSP v této bakalářské práci je samoorganizující se síť. Tento přístup byl poprvé představen v [1, 6] a následně byla publikována řada algoritmů založených na stejném samoorganizujícím principu. Algoritmus použitý v této práci vychází z přístupů [13, 14], které jsou podobně jako předchozí přístupy založeny na kompetitivním učení, které uspořádává neurony tak, aby jejich posloupnost aproximovala výslednou cestu. Toto kompetitivní učení je také známo jako Kohonenova mapa.



Obrázek 1.1: Schéma dvouvrstvé neuronové sítě. Množina $\mathbf{N} = \{n_1 \dots n_m\}$ reprezentuje vrstvu výstupních neuronů a množina $\mathbf{C} = \{c_1 \dots c_n\}$ je vrstva vstupních vektorů.

Samotná samoorganizující se síť se skládá ze dvou vrstev, které jsou navzájem propojené. První vrstva je tvořena množinou měst $\mathbf{C} = \{c_1 \dots c_n\}$, druhá vrstva je dána množinou neuronů $\mathbf{N} = \{n_1 \dots n_m\}$. Výsledná posloupnost adaptovaných neuronů z množiny $\mathbf{N}$ reprezentuje cestu pro obchodního cestujícího. Na obrázku 1.1 jsou zobrazeny zmiňované vrstvy neuronové sítě.

Síť je inicializována s malými vahami spojení mezi neurony a městy. Tyto váhy se během algoritmu mění a to tak, že pro každé náhodně vybrané město se vybere vítězný

[1] PAM vzdálenost (nebo také Dayhoff-Eckova vzdálenost či PAM hodnota) je vzdálenost mezi posloupností proteinů, která je definována jako minimální počet přijatých mutací na 100 aminokyselin, které jsou nutné k transformaci proteinu na jiný protein. Jedna PAM hodnota je jednotkou evoluce.

neuron. Tento vítězný neuron se následně adaptuje podle adaptační funkce a váha spojení mezi vítězným neuronem a městem se změní. Takto se postupně síť sama organizuje, dokud není splněno ukončovací kritérium.

Algoritmus 1: Algoritmus SOM

Vstup: $\mathbf{C} = \{c_1, c_2, \ldots c_m\}$ množina měst, kde $c_d \in \mathbf{C}$ je depot
Vstup: m je počet obchodních cestujících
Vstup: (G, μ, α) parametry samoorganizující se neuronové sítě
Vstup: δ maximální povolená chyba
Výstup: $(n_1, n_2, \ldots, n_n)$ neurony reprezentující cestu

```
1  begin
2      inicializace (n_1, n_2, ..., n_n)
3      while δ < chyba do
4          chyba ⟵ 0
5          I ⟵ ∅
6          ∏(C) ⟵ vytvoř náhodnou permutaci měst
7          foreach c ∈ ∏(C) do
8              v* ⟵ vyber vítězný neuron k městu c, v* ∉ I
9              adaptuj (v*, c)
10             chyba ⟵ max{chyba, | v*, c |}
11             I ⟵ I ∪ {v*}
12         G ⟵ (1 − α) · G
13 end
```

Vstupem algoritmu 1 je množina měst, mezi kterými se hledá cesta. Na začátku algoritmu jsou inicializovány neurony do prostoru, který je vymezen okolím depotu. Tyto neurony nesou informaci o svých sousedních neuronech (jeden neuron má právě dva sousední neurony), a proto spolu všechny neurony tvoří posloupnost, která je zde nazývána řetízek.

Aby některá města nebyla upřednostňována před ostatními městy, provede se na počátku každé iterace náhodná permutace měst a poté se nalezne pro každé město nejbližší povolený neuron. Následuje adaptační proces. V něm jsou vítězný neuron a jeho sousední neurony přibližovány k městu. Míra přiblížení je určena funkcí sousedských vztahů f. Adaptaci lze vyjádřit vztahem

$$n_j^\star = n_j + \mu \cdot f(G, l)|c_i, n_j|, \tag{1.4}$$

kde μ míra učení, n_j jsou souřadnice j-tého neuronu, $n_j^\star$ jsou nově adaptované souřadnice neuronu n_j, c_i je město, ke kterému jsou neurony přibližovány a $|\ldots|$ je Euklidovská vzdálenost. Zmiňovaná funkce sousedských vztahů je dána následujícím vztahem

$$f(G, d) = \begin{cases} e^{-\frac{d^2}{G2}} & d < 0.2M, \\ 0 & \text{jinde}, \end{cases} \tag{1.5}$$

kde G je parametr učení, l je délka řetízku tvořeného neurony a m je počet neuronů daný vztahem

$$m = 2n, \tag{1.6}$$

kde n je počet měst.

Z důvodu lepšího uspořádání neuronové sítě se pro daný iterační krok adaptované neurony zakážou a již je nelze vybrat jakožto vítězné neurony. Tímto způsobem se zajistí, že může být adaptováno větší množství neuronů a dojde tak k jejich lepšímu využití.

S postupující iterací se snižuje parametr G podle vztahu

$$G = (1 - \alpha) \cdot G, \tag{1.7}$$

kde α je parametr poklesu. Vhodná počáteční hodnota parametru G_0 byla experimentálně určena v [13] a funkce pro její určení je následující

$$G_0 = 12.41n + 0.06. \tag{1.8}$$

Po skončení algoritmu řetízek neuronů aproximuje cestu obchodního cestujícího. Ze získaného řetízku se určí posloupnost měst, která vyjadřuje pořadí, ve kterém mají být jednotlivá města navštívena.

Tabulka 1.1: Parametry SOM pro TSP

Parametr	**Popis parametru**
$m = 2n$	počet neuronů v řetízku
$G_0 = 12.41n + 0.06$	počáteční hodnota parametru přírustku
$d = 0.2M$	parametr pro funkci sousedních vztahů
$\mu = 0.6$	parametr učení
$\alpha = 0.1$	parametr poklesu
$\delta = 0.001$	maximální chyba

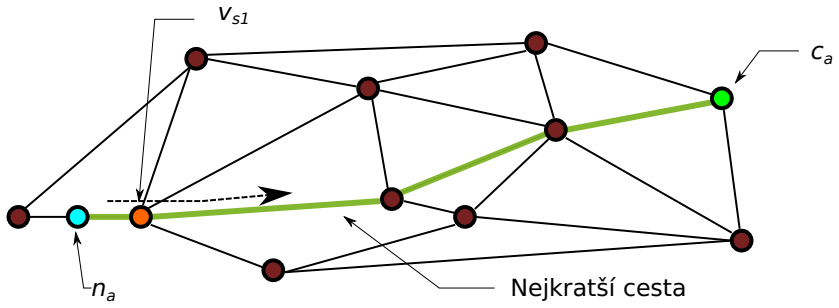
1.4 Samoorganizující se síť pro problém obchodního cestujícího s ne-Euklidovskou vzdáleností

Využitím samoorganizující se sítě pro řešení úlohy obchodního cestujícího v problému plánování cesty přes více cílů se zabývají autoři [3]. V tomto problému je vyžadováno spojení měst cestou, která nesmí protínat překážky, a proto nelze uvažovat Euklidovské vzdálenosti mezi městy. Pro učení sítě to znamená nahrazení Euklidovské metriky při výběru vítězného neuronu délkou nejkratší cesty mezi neuronem a daným městem. Podobně ve fázi adaptace musejí neurony respektovat nejkratší cesty. Vlastní adaptaci si tak lze představit jako pohyb neuronů podél příslušných nejkratších cest k danému městu, přičemž neuron urazí na cestě vzdálenost odpovídající funkci sousedských vztahů.

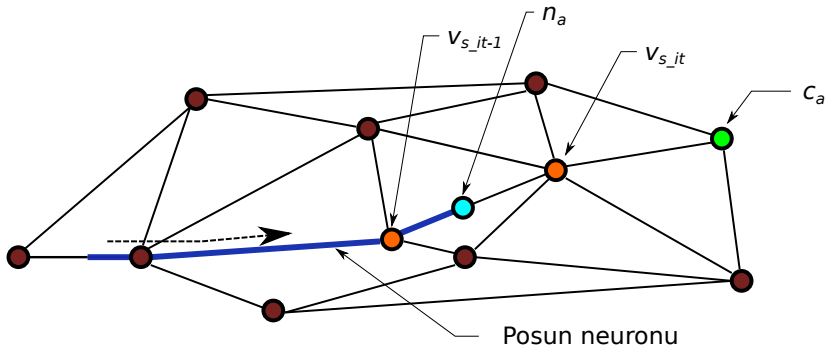
Hlavní problém využití samoorganizujících se sítí v problému hledání cesty přes více cílů je výpočetní náročnost určení nejkratší cesty. V [3] autoři proto použili jednoduchou aproximaci založenou na konvexním rozdělení volného prostoru reprezentovaného polygonální doménou na množinu konvexních vrcholů. Neuronny jsou tedy vždy v nějaké konvexní oblasti a hrubý odhad nejkratší cesty je tak určen jako nejkratší cesta přes některý z vrcholů oblasti, které jsou zároveň vrcholy polygonální domény.

1.5 Samoorganizující se síť na grafu

Alternativou k [3] je zmapování volného prostoru grafem. Tento přístup poprvé představil Takeshi Yamakawa [15]. V tomto případě se neurony pohybují po grafu a je tak zajištěno, že se nikdy nebudou nacházet uvnitř překážky a že vzdálenost mezi neuronem a městem bude determinována volným prostorem. Města určená k navštívení jsou reprezentována určitými vrcholy grafu, a proto lze najít vždy cestu z jakéhokoliv vrcholu do libovolného města (musí být splněno, že graf obsahuje právě jednu komponentu souvislosti se všemi městy). Vítězný neuron je určen jako ten neuron, jenž má minimální vzdálenost k danému městu na grafu.



(a) vítězný neuron před adaptací



(b) vítězný neuron po adaptaci

Obrázek 1.2: Obrázek (a) zobrazuje schéma před adaptací neuronu n_a k městu c_a. Zeleně je vyjádřena nejkratší cesta na grafu, šipka ukazuje směr adaptace. Na obrázku (b) má neuron n_a již nové souřadnice určené ze vztahu $n_a = v_{s\_it-1} + \beta \cdot |v_{s\_it}, v_{s\_it-1}|$, kde v_{s_it} a v_{s_it} jsou vrcholy grafu a $|\ldots|$ je Euklidovská vzdálenost.

Neurony se vždy nachází na nějaké hraně grafu, proto je nutné při určování cesty mezi neuronem n a městem c určit dvě délky cest d_1 a d_2

$$d_1 = |n, c|_i = |n, v_i| + |v_i, c|, \tag{1.9}$$

$$d_2 = |n, c|_j = |n, v_j| + |v_j, c|, \tag{1.10}$$

kde v_i a v_j jsou vrcholy incidentní s neuronem n. Při určování vítězného neuronu $n^\star$ se vychází z délky d, která je dána vztahem

$$d = \min\{d_1, d_2\}. \tag{1.11}$$

Vítězný neuron je určen ze vztahu

$$n^\star \longleftarrow |n^\star, c| = d. \tag{1.12}$$

Získaný vítězný neuron je následně adaptován na grafu. Pro adaptaci je nutné zajistit posloupnost vrcholů $V_s = \{v_{s1}, v_{s2}, \ldots, v_{sm}\}$, která je dána vrcholy grafu, jež jsou obsaženy v nejkratší cestě mezi vítězným neuronem $n^\star$ a městem c. Dále je nutný seznam vzdáleností W.

Algoritmus 2: Adaptační algoritmus na grafu

Vstup: c_a - město, ke kterému bude adaptován neuron
Vstup: n_a - neuron, který bude adaptován
Vstup: $V_s = \{v_{s1}, v_{s2}, \ldots, v_{sm}\}$ - posloupnost seřazených vrcholů reprezentujících cestu
Vstup: $W = \{w_1, w_2, \ldots, w_m\}$ - seznam ohodnocení hran mezi vrcholy z V_s
Vstup: β - parametr určující míru adaptace
Výstup: $n_a^\star$ - neuron n_a s novými souřadnicemi

```
1  begin
2      vzdál_celk ⟵ |n_a, c_a|
3      vzdál_adapt ⟵ β·vzdál_celk
4      vzdál_tmp ⟵ 0
5      for it ← 0 to m do
6          vzdál_tmp ⟵ vzdál_tmp + w_i
7          if vzdál_adapt < vzdál_tmp then
8              β_relativ ⟵ (vzdl_adapt − (vzdl_tmp − w_it / w_it))
9              if it = 0 then
10                 n_a^⋆ ⟵ n_a + β_relativ · |v_s_it, n_a|
11                 break
12             else
13                 n_a^⋆ ⟵ v_s_it−1 + β_relativ · |v_s_it, v_s_it−1|
14                 break
15 end
```

Tento seznam obsahuje na prvním místě vzdálenost mezi neuronem $n^\star$ a prvním vrcholem v_{s1} z množiny V_s, na dalších místech seznamu jsou vzdálenosti mezi sousedícími vrcholy z posloupnosti V_s. Zde je matematický vztah pro určení W

$$W = \{|n^\star, v_{s1}|, |v_{s1}, v_{s2}|, \ldots, |v_{sm-1}, c_a|\}, \tag{1.13}$$

$$\text{kde } c_a = v_{sm}.$$

Pro určení nových souřadnic adaptovaného neuronu n_a se využije algoritmus 2. Tento algoritmus postupně prochází seznam V_s a počítá vzdálenost $vzdál_{tmp}$ od vítězného neuronu $n^\star$ k aktuálnímu vrcholu v_{s_it}. Pokud je vzdálenost $vzdál_{tmp}$ větší než vzdálenost

vzdál$_{adapt}$, o kterou se má neuron posunout na grafu, vypočítají se nové souřadnice neuronu $n^\star$ za pomocí vrcholů v_{s_it} a v_{s_it-1}.

Existuje specifický případ tohoto algoritmu, který je nutno ošetřit. A to pokud se souřadnice adaptovaného neuronu n_a budou vyskytovat na hraně mezi neuronem $n^\star$ a v_{s1}. V tomto případě se nové souřadnice adaptovaného neuronu určí z souřadnic v_{s_it} a $n^\star$. Kompletní vztahy pro výpočet nových souřadnic adaptovaného neuronu jsou v algoritmu 2 a to na řádcích 10 a 13.

Kapitola 2

Neuronové sítě pro reprezentaci polygonální domény

Hlavním problémem pro plánování cesty přes více cílů je vhodné zvolení reprezentace volného prostoru. Pro reprezentaci volného prostoru byla vybrána polygonální doména. Pokud se obchodní cestující pohybuje v polygonální doméně, je zajištěno, že se nikdy nebude nacházet uvnitř překážky a ani cesty mezi městy nebudou překážky protínat. Všechna města musí být samozřejmě uvnitř polygonální domény, čímž je zaručeno, že bude-li se obchodní cestující pohybovat na polygonální doméně, najde vždy cestu k městu. Kromě grafu viditelnosti a grafu trojúhelníkové mřížky byly pro reprezentaci polygonální domény zvoleny i grafy generované neuronovými sítěmi.

Neuronové sítě jsou schopny algoritmicky zpracovávat kvantizované vektory, které popisují libovolnou varietu vstupních dat, jež mají být zpracována. Varieta je popsána určitým počtem neuronů se samoorganizujícími se schopnostmi, takže popis variety se postupně zlepšuje. Mezi těmito neurony vznikají spojení, které popisují jednotlivé vztahy mezi neurony a dále tak přispívají ke kvalitnějšímu popisu.

Technika vektorové kvantizace přibližuje varietu vstupů $M \subseteq R^n$ díky konečné množině A skládající se z vektorů tzv. referencí $w_i \in R_n, i = 1, 2..., N$. Dále se zde využívá datový vektor $v \in M$, ten je popsán tzv. vítěznou referencí w_x z množiny A, pro kterou platí, že chyba vektoru v a w_x je minimální (v našem případě je chyba reprezentována vzdáleností). Tímto rozdělíme varietu do určitého počtu subregionů

$$M_x = \{v \in M \mid\mid\mid v - w_x \mid\mid \leq \mid\mid v - w_i \mid\mid, \forall i\},$$

které se také nazývají Voronoiovy polygony, proto každý datový vektor v je popsán odpovídajícím referenčním vektorem w_x.

V následujících oddílech kapitoly je nejprve definováno kompetitivní Hebbovo učení, které je dále využíváno při tvorbě grafů neuronového plynu a rostoucího neuronového plynu. Algoritmy vytvářející tyto grafy jsou posány blíže a v závěru kapitoly jsou představeny navrhnuté přístupy, jimiž se tyto algoritmy modifikovaly tak, aby je bylo možné použít pro reprezentaci polygonální domény jakožto vstupu algoritmu SOM pro TSP.

2.1 Kompetitivní Hebbovo učení

Hebbovo učení vychází ze základů biologických neuronových sítí. Jeho myšlenkou je posilování synapse neuronů, které mají stejné nebo podobné stavy (charakteristiky) a naopak omezení synapse mezi neurony se stavy odlišnými. V našem případě jde o vytváření spojení mezi blízkými neurony a zároveň potlačení vzniku spojení mezi vzdálenými neurony, čímž je docíleno získání výsledných topologických vztahů mezi neurony.

Kompetitivní Hebbovo učení obvykle neslouží ke konečnému popisu vstupních dat, i když i to je možné, ale spíše funguje jako pojící složka složitějších algoritmů, nebo se využívá při zkoumání chování neuronových sítí.

Princip je následující. Nejprve je inicializována množina neuronů. Souřadnice těchto neuronů náhodně popisují množinu vstupních dat. V každém iteračním kroku je náhodně vybrán vstupní bod, pro který je splněno, že $P(\xi) > 0$ (což znamená, že pravděpodobnost výskytu vstupního bodu v množině vstupních dat je větší než nula) a pro tento bod jsou určeny dva nejbližší neurony. Jestliže mezi těmito neurony neexistuje spojení, tak se toto spojení vytvoří.

Je důležité poznamenat, že kompetitivní Hebbovo učení nijak při průběhu algoritmu nemění souřadnice neuronů, pouze se snaží najít optimální sousedské vztahy mezi neurony. Algoritmus kompetitivního Hebbova učení je popsán následovně.

Nechť množina $\mathbf{D}$ reprezentuje vstupní data.

1. Inicializuj množinu $\mathbf{N}$, která obsahuje požadovaný počet neuronů, jenž jsou náhodně inicializovány podle množiny vstupních dat $\mathbf{D}$.
2. Inicializuj množinu spojů $\mathbf{C}$, kde $\mathbf{C} \subseteq \mathbf{N} \times \mathbf{N}$ jako prázdnou množinu: $\mathbf{C} = \emptyset$.
3. Náhodně vyber prvek (bod) p z množiny vstupních dat $\mathbf{D}$.
4. Nalezni vítězný neuron s_1 a druhý nejbližší neuron s_2 k bodu p tak, aby platilo

$$s_1 = \text{argmin}_{n \in \mathbf{N}} |p, n_i| \tag{2.1}$$

   a

$$s_2 = \text{argmin}_{n \in \mathbf{N} \setminus s_1} |p, n_j|. \tag{2.2}$$

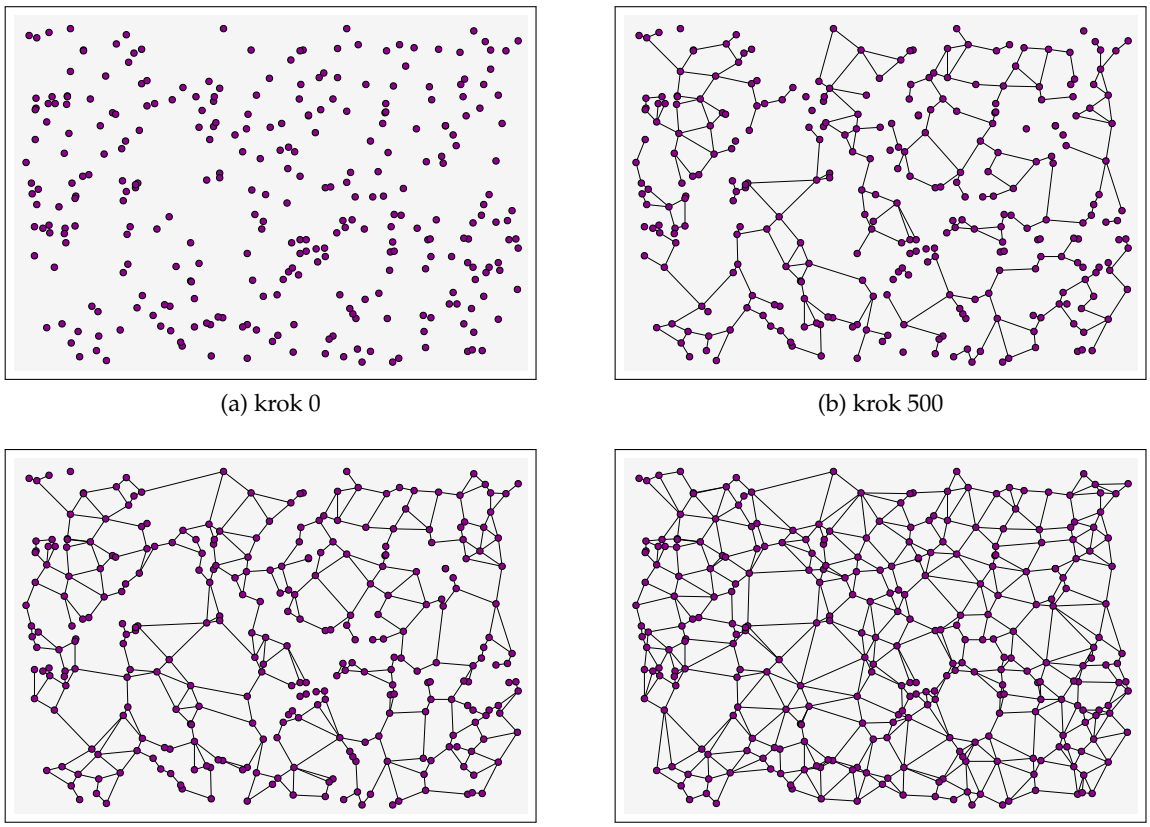
5. Jestliže neexistuje spojení mezi s_1 a s_2, tak jej vytvoř

$$\mathbf{C} = \mathbf{C} \cup \{(s_1, s_2)\}. \tag{2.3}$$

6. Pokud nebylo dosaženo maximálního počtu iterací, vrať se na krok 3.

2.2 Síť neuronového plynu

Neuronový plyn je neuronová samoorganizující se síť představená Thomasem Martinetzem a Klausem Schultenem [9]. Jde o algoritmus pro nalezení reprezentace vstupních dat. Jeho pojmenování je odvozeno od chování reálného plynu v prostoru, který tento algoritmus připomíná.



(a) krok 0 (b) krok 500

(c) krok 1000 (d) krok 5000

Obrázek 2.1: Utváření kompetitivnho Hebbova učení v jednotlivých krocích.

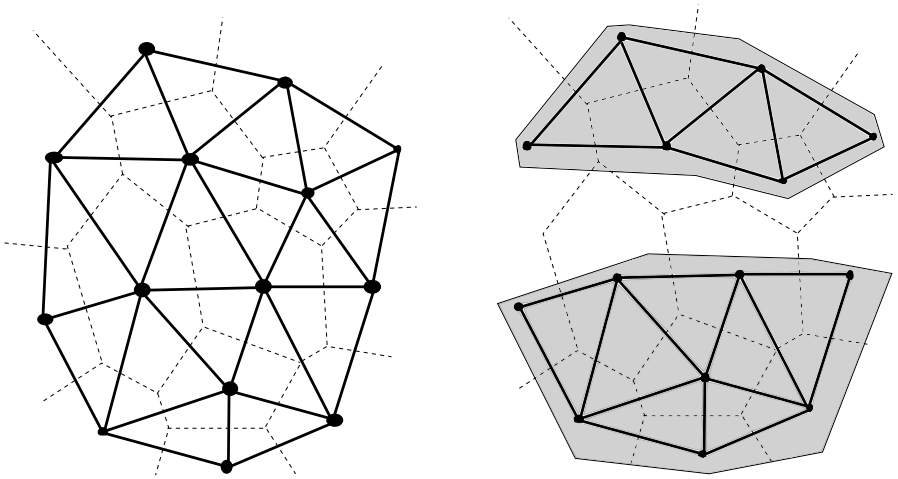
2.2.1 Kompetitivní Hebbovo učení a algoritmus neuronového plynu

Kompetitivní Hebbovo učení předpokládá daný počet neuronů v $\mathbf{R}^n$. Mezi těmito neurony postupně vznikají topologická spojení, která reprezentují vstupní body. Ve studovaném problému platí, že pro každý vstupní bod x jsou spojeny hranou dva nejbližší neurony (měřeno v Euklidovské vzdálenosti). Výsledný graf je podgraf tzv. Delaunayovy triangulace[1] zobrazené na obrázku 2.2a, která odpovídá množině neuronů. Tento podgraf nazývaný indukovaná Delaunayova triangulace na obrázku 2.2b je vymezen vstupním prostorem, pro který platí $P(\xi) > 0$.

Pouze neurony ležící na vstupních datech nebo jejich okolích dále rozvíjí hrany. Ostatní neurony jsou nepotřebné a jsou nazývány mrtvé jednotky (death units). Pro zajištění využitelnosti všech neuronů se musí neurony nacházet tam, kde platí $P(\xi) > 0$. Toho může být docíleno použitím vektorové kvantizace. Řešením je algoritmus neuronového plynu, jehož hlavní princip využívá přiblížení k nejbližších neuronů ke vstupnímu bodu x, kde míra přiblížení klesá s hodnotou k. Vysoká počáteční hodnota k způsobuje velké posunutí neuronů ke vstupnímu bodu. Parametr k postupně klesá, dokud se ke vstupnímu bodu neposunuje pouze jeden neuron.

Pro daná vstupní data lze nejdříve aplikovat princip, kdy jsou neurony přizpůsobovány ke vstupním bodům a následně vznikají topologická spojení mezi neurony podle

[1]Delaynayova triangulace [11] je taková triangulace, která maximalizuje minimální úhel všech trojúhelníků triangulace.



(a) Delaunayova triangulace

(b) indukovaná Delaunayova triangulace

Obrázek 2.2: Obrázek (a) zobrazuje Delaunayovu triangulaci (silná čára), spojující body, jejichž Voronoiovy polygony (čárkovaná čára) spolu sousedí. Na obrázku (b) je indukovaná Delaunayova triangulace (silné čáry), která se indukuje pouze na prostoru vstupních dat (šedá oblast), tj. tam kde $P(\xi) > 0$.

kompetitivního Hebbova učení. Obě tyto techniky kombinuje algoritmus neuronového plynu. Při současném použití obou technik se objevuje otázka aktuálnosti (správnosti) hran mezi jednotlivými neurony. Z důvodu, že se neurony různě pohybují, nemusí spojení dříve vytvořené již správně reprezentovat vstupní prostředí a je tak nutno toto spojení smazat. To je vyřešeno metodou stárnutí hran, kdy každá hrana má své stáří, které je aktualizováno a pokud toto stáří hrany přesáhne určitou hodnotu, hrana je smazána.

2.2.2 Algoritmus neuronového plynu

Algoritmus neuronového plynu je samoorganizující se neuronová síť založená na Kohonenových mapách. Neurony jsou postupně adaptovány podle vstupních bodů, které náhodně reprezentují vstupní prostor. Tyto neurony se poté rovnoměrně rozprostřou po vstupním prostoru a samy tak mohou být považovány za reprezentaci vstupního prostoru.

Na začátku algoritmu jsou inicializovány samotné neurony reprezentující vstupní data. Neurony jsou náhodně rozmístěny v prostoru, který má být zmapován. Dále je inicializována množina spojení mezi neurony, tj. ke každému spojení je přiřazena hodnota nula (tzn. spojení mezi danými neurony neexistuje).

Následující postup se iteračně opakuje, dokud není splněna podmínka ukončení. Náhodně se vybere vstupní bod (musí být zajištěno, že pravděpodobnost výběru je pro každý vstupní bod stejná) a pro každý neuron se určí vzdálenost neuronu k vstupnímu bodu. Nyní se všechny neurony setřídí vzestupně podle získaných vzdáleností. Pro takto setříděné neurony se postupně modifikují jejich souřadnice podle

$$w_i^{new} = w_i^{old} + \varepsilon(t) h_\lambda(k_i(v, A)) \cdot |v, w_i|, \tag{2.4}$$

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{\frac{t}{t_{max}}}, \tag{2.5}$$

$$\varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{\frac{t}{t_{max}}}, \tag{2.6}$$

$$h_\lambda(k) = e^{\frac{-k}{\lambda(t)}}, \tag{2.7}$$

kde w_i jsou souřadnice *i-tého* neuronu, t je aktuální iterace, t_{max} maximální počet iterací, k je pořadí w_i v seřazeném seznamu neuronů, v je vstupní bod a $|\dots|$ je Euklidovská vzdálenost.

Z rovnice (2.4) je patrné, že nejbližší neuron se nejvíce přiblíží ke vstupnímu bodu, zatímco s inkrementujícím se pořadím seřazených neuronů exponenciálně klesá délka posuvu daného neuronu ke vstupnímu bodu.

Algoritmus 3: Algoritmus neuronového plynu

Vstup: $\boldsymbol{D} = \{d_1, ..., d_n\}$ - množina vstupních dat
Vstup: N - maximální počet neuronů
Vstup: it_{max} - maximální počet iterací
Vstup: $(\lambda_i, \lambda_f, \varepsilon_i, \varepsilon_f, t_{max})$ - parametry neuronového plynu
Výstup: $\mathbf{N} = \{n_1, ..., n_N\}$ - množina neuronů
Výstup: $\mathbf{C} = \{\{c_{1,1}, ..., c_{1,N}\}...\{c_{1,N}, ..., c_{N,N}\}\}$ - množina spojeni

```
1  begin
2      inicializace množiny neuronů N
3      inicializace množiny spojení C
4      inicializace množiny stáří spojení T
5      it ⟵ 0                                   // it určuje aktuální iteraci
6      while it < it_max do
7          d* ⟵ náhodně vybraný prvek z množiny D
8          pro každý neuron n se určí vzdálenost |d*, n|
9          vzestupně se setřídí neurony podle vzdálenosti k d*
10         k_i určuje pořadí neuronu v seřazené sekvenci neuronů
11         foreach n_i ∈ N do
12             n_i^new ⟵ n_i^old + ε(t)h_λ(k_i(d*, D)) · |d*, n_i|
13             if c_{i0 i1} = 0 then
14                 c_{i0 i1} ⟵ 1
15                 T_{i0 i1} ⟵ 0
16             else
17                 T_{i0 i1} ⟵ 0
18         T_{i0 i_j} ⟵ T_{i0 i_j} + 1 pro všechna j, pro které je c_{i0 i_j} = 0
19         c_{i0 i_j} ⟵ 0,  ∀j, pro něž platí: c_{i0 i_j} = 0 & T_{i0 i_j} > t_max
20         it ⟵ it + 1
21 end
```

Dále je nutné zajistit utváření vztahů mezi neurony a to tak, aby co nejlépe popisovaly vstupní data. Proto se vždy spojí dva neurony (pokud již tomu tak není), které jsou nejblíže k vybranému vstupnímu bodu a zároveň se nastaví stáří jejich spojení na nulu.

Tím je vyjádřena aktuálnost spojení (čím menší, tím aktuálnější). Následně se inkrementuje stáří spojení pro každý neuron spojený s neuronem, který je nejblíže vybranému vstupnímu bodu. S přibývající iterací neurony mapují stále lépe a lépe prostředí, ale některá spojení mezi neurony nemusí být již aktuální. Proto se na konci každého iteračního

kroku smažou spojení, jejichž stáří přesáhne povolenou maximální hodnotu t_{max}. Algoritmus neuronového plynu je schématicky popsán algoritmem 3. Parametry algoritmu neuronového plynu jsou v tabulce 2.1, jejichž hodnoty byly převzaty z [8].

Tabulka 2.1: Parametry algoritmu neuronového plynu

Parametry			
λ_i	λ_f	ϵ_i	ϵ_f
10	$0,01$	$0,5$	$0,05$

2.3 Síť rostoucího neuronového plynu

Síť rostoucího neuronového plynu (GNG) představená B. Fritzkem [7] je inspirovaná algoritmem rostoucího plynu. Hlavní odlišností algoritmů je to, že GNG na počátku inicializuje pouze dva neurony a s přibývající iterací se počet neuronů mění (typicky se jejich počet zvětšuje) na základě ohodnocování lokálního statistického měření získaného v předchozích iteračních krocích. Tento mechanismus rozrůstající se množiny neuronů s modifikací grafu neuronové sítě využívající kompetitivního Hebbova učení popíše vstupní data $n-$dimenzionálním uspořádání neuronů.

Adaptační proces je dán vybíráním dvou nejbližších neuronů ke vstupnímu bodu a následným přiblížením těchto neuronů k tomuto bodu. Rozrůstání sítě je zajištěno přidáváním nových neuronů po určitých intervalech iterace.

Pokud se vyskytnou nějaké neurony odtržené od indukované Delaunayovy triangulace, je nutné je smazat. Toho je dosaženo lokálním stárnutím hran, které spojují nejbližší neuron ke vstupnímu bodu se sousedními neurony. S přidáváním či odebíráním neuronů se síť přibližuje indukované Delaunayově triangulaci, která je pomalu adaptována v závislosti na množině vstupních dat.

Akumulace čtverce vzdálenosti během adaptačního procesu napomáhá k identifikaci neuronů ležících ve vstupním prostoru, ve kterém mapování vstupních bodů způsobuje velkou lokální chybu neuronů. Přidáním nových neuronů do prostoru se tato chyba sníží. Algoritmus rostoucího neuronového plynu lze popsat následovně.

Nechť množina $\mathbf{D}$ reprezentuje vstupní data a $\alpha, \beta, \varepsilon_b, \varepsilon_n, t_{max}$ jsou parametry algoritmu GNG.

1. Inicializuj množinu $\mathbf{N}$, která obsahuje dva náhodně inicializované neurony n_1, n_2, kde $\mathbf{N} = \{n_1, n_2\}$.
2. Inicializuj množinu spojů $\mathbf{C}$, kde $\mathbf{C} \subseteq \mathbf{N} \times \mathbf{N}$ jako prázdnou množinu $\mathbf{C} = \emptyset$.
3. Náhodně vyber prvek (bod) p z množiny vstupních dat $\mathbf{D}$.
4. Nalezni vítězný neuron s_1 a druhý nejbližší neuron s_2 k bodu p tak, aby platilo

$$s_1 = \text{argmin}_{n \in \mathbf{N}} |p, n_i| \tag{2.8}$$

a

$$s_2 = \text{argmin}_{n \in \mathbf{N} \setminus s_1} |p, n_j|. \tag{2.9}$$

5. Jestliže neexistuje spojení mezi s_1 a s_2, tak jej vytvoř

$$\mathbf{C} = \mathbf{C} \cup \{(s_1, s_2)\}. \tag{2.10}$$

Zároveň vynuluj stáří spojení mezi s_1 a s_2

$$t_{(s_1,s_2)} = 0. \tag{2.11}$$

6. Přidej čtverec vzdálenosti mezi bodem p a nejbližším neuronem s_1 k lokální chybě s_1

$$E_{s_1} = E_{s_1} + |p - s_1|^2. \tag{2.12}$$

7. Adaptuj souřadnice neuronu s_1 a jeho topologických sousedů za pomocí parametru ε_b respektivě ε_n

$$s_1 = s_1 + \varepsilon_b |p, s_1|, \tag{2.13}$$

$$s_1 = s_1 + \varepsilon_n |p, s_i|, \tag{2.14}$$

$$\text{pro } \forall i \in \mathbf{S_{s_1}},$$

kde $\mathbf{S_{s_1}}$ je množina všech topologických sousedů s_1.

8. Navyš stáří spojení pro každou hranu jdoucí z s_1

$$t_{(s_1,i)} = t_{(s_1,i)} + 1, \text{ pro } \forall i \in \mathbf{S_{s_1}}. \tag{2.15}$$

9. Odeber všechny hrany, které mají stáří hrany větší než t_{max}. Pokud dojde k tomu, že neuron nebude mít již žádné topologické sousedy, pak smaž i tento neuron.

10. Pokud je počet iterací celočíselně dělitelný bezezbytku parametrem λ, tak vytvoř nový neuron podle následujícího schématu:

- Urči neuron q, který má maximální lokální chybu

$$s_q = \text{argmax}_{n \in \mathbf{N}} E_n. \tag{2.16}$$

- Ze všech sousedů s_q urči ten neuron, pro jehož maximální lokální chybu platí

$$s_f = \text{argmax}_{n \in \mathbf{S_q}} E_n, \tag{2.17}$$

kde $\mathbf{S_{s_q}}$ je množina všech topologických sousedů s_q.

- Přidej do množiny neuronů $\mathbf{N}$ nový neuron n_{new} se souřadnicemi určenými středem úsečky definované neurony s_q a s_f

$$n_{new} = \frac{n_q + n_f}{2}, \tag{2.18}$$

$$\mathbf{N} = \mathbf{N} \cup \{n_{new}\}. \tag{2.19}$$

- Do množiny spojení přidej nová spojení definována neurony n_{new} a n_q respektivě n_{new} a n_f. Zároveň odstraň spojení mezi n_q a n_f

$$\mathbf{C} = \mathbf{C} \cup \{(n_{new}, n_q), (n_{new}, n_f)\} \setminus \{(n_q, n_f)\}. \tag{2.20}$$

- Sniž lokální chybu E_{n_q} a E_{n_f} parametrem α

$$E_{n_q} = E_{n_q} - \alpha E_{n_q}, \tag{2.21}$$

$$E_{n_f} = E_{n_f} - \alpha E_{n_f}. \tag{2.22}$$

- Interpoluj lokální chybu neuronu n_{new} z lokálních chyb neuronů n_f a n_f

$$E_{n_{new}} = \frac{E_{n_f} + E_{n_f}}{2}. \tag{2.23}$$

11. Sniž lokální chyby všech neuronů

$$E_n = E_n - \beta E_n, \tag{2.24}$$

$$\text{pro } \forall n \in \mathbf{N}.$$

12. Pokud není splněno kritérium ukončení, vrať se na krok 3.

Ukončovací pravidlo může být de facto libovolné, algoritmus může skončit po dosažení určitého počtu iterací, po přidání daného počtu neuronů, či po překročení určité maximální chyby, nebo mohou být tyto principy mezi sebou kombinovány. Tabulka 2.2 zobrazuje hodnoty parametrů GNG převzaté z [10].

Tabulka 2.2: Parametry algoritmu GNG

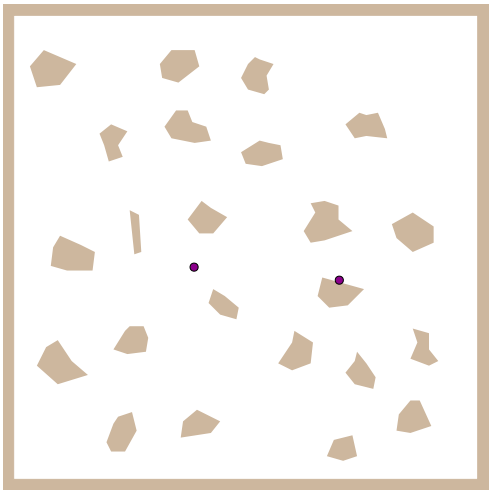
Parametry				
λ	ϵ_b	ϵ_n	α	β
600	$0,05$	$0,0006$	$0,05$	$0,0005$

Pro grafovou reprezentaci volného prostoru byly naimplementovány oba algoritmy a to algoritmus neuronového plynu a algoritmus GNG. Výsledná grafová reprezentace je pro oba algoritmy obdobná, ale generace grafu algoritmem neuronového plynu trvala příliš dlouho. Je to z důvodu časové složitosti algoritmu. Asymptoticky nejnáročnější na celém algoritmu neuronového plynu je seřazování neuronů podle vzdálenosti k zrovna vybranému vstupnímu bodu, které má asymptotickou složitost $N\log_2 N$. Výsledná asymptotická složitost celého algoritmu je $N^2\log_2 N$, kde N je počet neuronů. Protože počet iteračních kroků pro hustší sítě dosahoval statisíců, trval běh algoritmu příliš dlouho.

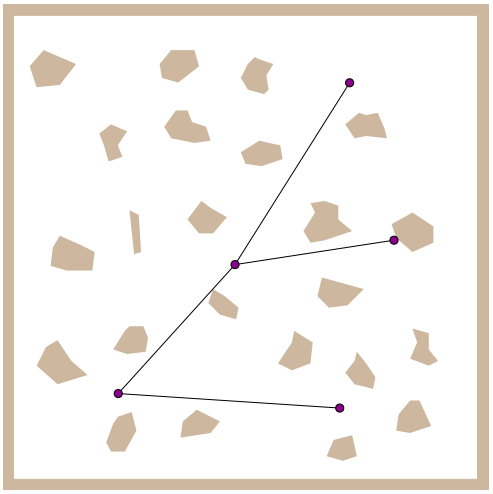
I když algoritmus GNG potřebuje k vytváření grafu daleko více iteračních kroků než algoritmus neuronového plynu, generace grafu netrvá tak dlouho. Proto bylo rozhodnuto, že algoritmus neuronového plynu nebude dále rozvíjen a pro reprezentaci polygonální domény se využije pouze graf GNG.

2.4 Inicializace neuronů GNG v závislosti na TSP

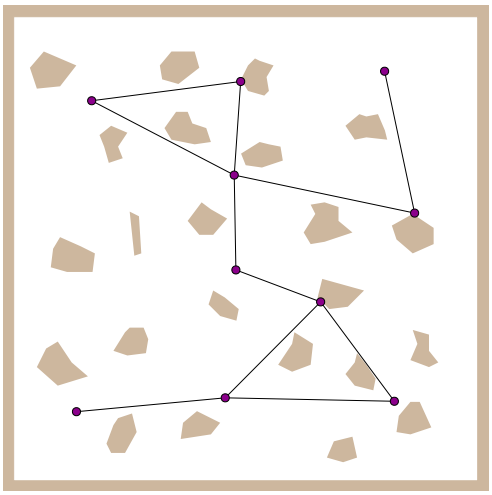
Vstupem algoritmu obchodního cestujícího jsou města, proto má-li být využit graf jako reprezentace volného prostoru, musí být do tohoto grafu města zakomponována. Pro inicializaci měst byl navržen následující přístup. Ten na rozdíl od původní inicializace dvou



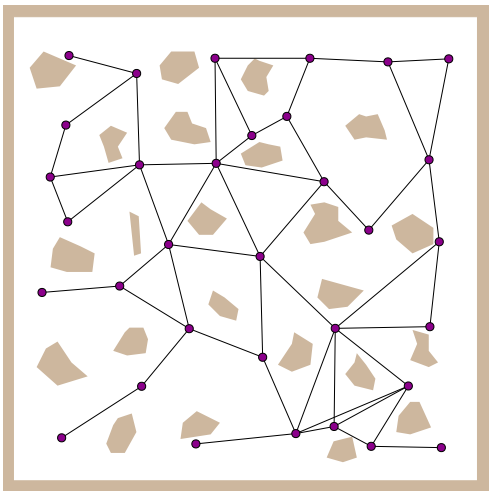
(a) krok 0



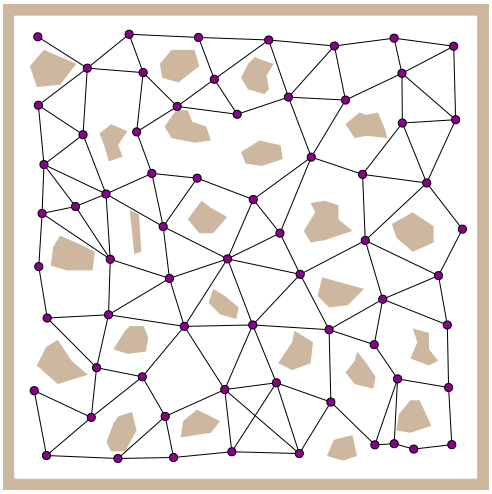
(b) krok 3000



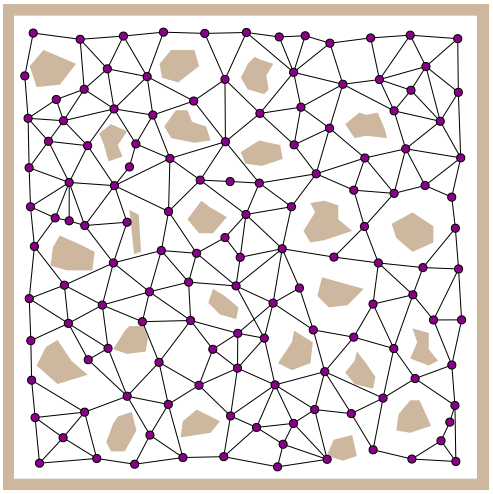
(c) krok 6000



(d) krok 20 000



(e) krok 40 000



(f) krok 80 000

Obrázek 2.3: Grafické zobrazení vývoje grafu GNG.

neuronů v algoritmu GNG inicializuje množinu neuronů o velikosti počtu prvků množiny měst. Každý tento neuron reprezentuje právě jedno město i s jeho souřadnicemi. Aby bylo zajištěno, že neurony reprezentující města nebudou během iterace měnit své souřadnice, jsou všechny neurony (inicializované na počátku i ty později vytvořené) rozděleny do dvou skupin. A to na skupinu reprezentující města, jejichž souřadnice neuronů nesmí být jakkoliv měněny a na skupinu neuronů, jež mohou své souřadnice libovolně měnit.

2.5 Rozšíření GNG pro polygonální doménu

V reálných prostorech se mohou vyskytovat překážky. Má-li být popsán prostor s překážkami algoritmem GNG, není původní algoritmus GNG na takovýto problém připraven, proto musí být vylepšen.

Přístupů, jak se vypořádat s překážkami je více, a proto bylo navrženo několik algoritmů (algoritmus A, algoritmus B, algoritmus C, algoritmus D). Všechny tyto algoritmy jsou experimentálně vyzkoušeny v kapitole 3.

- Algoritmus A se nejvíce bliží filozofii algoritmu GNG a to tak, že počítá s tím, že všechny vstupní body, které formují graf, se budou nacházet pouze ve volném prostoru. To samo o sobě nezaručí, že se neurony (vrcholy) grafu nebudou občas nacházet vně překážky, což ale během algoritmu není chyba. Z důvodu tohoto jevu je možné prostor kvalitněji zmapovat, protože při počáteční fázi algoritmu, kdy je v síti málo neuronů, je umožněno lépe rozprostřít neurony po celé mapě. S postupující iterací přibývá neuronů, čímž se zhušťuje síť a nejbližší možný neuron ke vstupnímu bodu není příliš daleko, a proto je přibližovací proces jemnější. Protože vstupní body jsou pouze z volného prostoru, neurony postupně opouštějí překážky a mapují samotný volný prostor určený vstupními daty.

  Po skončení iterace, kdy už neurony nebudou dále adaptovány, musí být nutně neurony (i s jejich hranami), které se nacházejí uvnitř překážek, smazány (čím užší překážka, tím větší pravděpodobnost, že se na ní vyskytnou nechtěné neurony). Dále musí být zajištěno vymazání všech hran, které protínají jakoukoliv překážku.

- Narozdíl od algoritmu A, jsou při iteraci algoritmu B vybírány i vstupní body, jejichž souřadnice se nacházejí i vně překážek. Tímto přístupem se ignorují překážky a de facto celou mapu pokryje síť neuronů. Po skončení vlastního algoritmu GNG musí být smazány všechny neurony nacházející se vně překážek a také i hrany protínající překážky.

- Dalším přístupem je algoritmus C. Jedná se o modifikaci algoritmu B. V algoritmu C platí, že vybraný neuron (určený k adaptaci) nesmí adaptovat své souřadnice vně překážku. Pokud by k takovéto adaptaci mělo dojít, neuron jednoduše své souřadnice nezmění. Mohlo by se zdát, že tento přístup zajišťuje to, že se vrcholy nebudou nacházet v místech překážek, tudíž není nutná kontrola neuronů vyskytujících se uvnitř překážek jako v algoritmu A a B. Avšak nesmí se zapomínat, že do sítě jsou postupně přidávány nové neurony, které se mohou tvořit i na překážce. Proto i zde se musí odfiltrovat špatně umístěné vrcholy.

- Poslední algoritmus pro vyhýbání se překážkám je algoritmus D. Tento algoritmus vychází z algoritmu A a to tak, že se snaží zabránit protínání překážek hranami sítě

již během algoritmu. To je zajištěno tím, že je zakázáno při utváření sítě během iterace jakkoliv vytvořit hranu mezi neurony, která by protínala překážku. V případě, že by se neurony pohnuly tak, že by jejich spojující hrana protínala překážku, byla by tato hrana smazána.

2.6 Strategie pro ukončováni GNG

Pro zajištění požadované hustoty a rozložení vrcholů grafu po celém volném prostoru, byly navrženy následující přístupy. První přístup před spuštěním iterace algoritmu GNG rovnoměrně vloží do volného prostoru tzv. kontrolní body (check points). Mezery mezi těmito body ve směru osy x jsou určeny procentuálním zlomkem per velikosti mapy na ose x. Obdobně jsou mezery určeny pro osu y.

Během běhu algoritmu, po uplynutí určitého počtu iteračních kroků, se opakovaně kontroluje, jestli každý kontrolní bod má ve svém okolí alespoň jeden neuron grafu. Okolí je dáno poloměrem r se středem v kontrolním bodě. Poloměr je dán vztahem

$$r = \frac{\varrho_x + \varrho_x}{1,5}, \tag{2.25}$$

kde ϱ_x či ϱ_y jsou vzdálenosti mezi sousedícími kontrolními body na dané ose.

Pokud pro všechny kontrolní body platí, že ve svém okolí mají alespoň jeden neuron grafu, může být algoritmus GNG ukončen. Pro lepší výsledek a kvůli správnému zakomponování nově přidaných neuronů se v iteraci určitou dobu pokračuje, přitom je zakázáno přidávání nových neuronů do sítě a je nastaven pevný maximální počet iterací vztahem

$$max_{steps} = \xi \cdot it, \tag{2.26}$$

kde it je aktuální iterační krok.

Parametr ξ je závislý i na parametru λ z původního algoritmu GNG a to tak, že čím větší je λ, tím více se ξ blíží k jedné včetně. Pro parametr ξ byla zvolena hodnota $1,1$, která vede na dostatečně kvalitní doběhnutí algoritmu.

Pro případy, kdy je požadována vysoká hustota neuronů grafu a tudíž je i velká hustota kontrolních bodů, je přednastaven parametr, max_{steps} na dostatečně velkou hodnotu tak, aby při přehnané hustotě kontrolních bodů netrvala iterace algoritmu GNG příliš dlouho.

Další dva navržené přístupy nevyužívají dynamického ukončování algoritmu GNG a místo toho před začátkem algoritmu nejprve určí maximální počet neuronů grafu a tím i maximální možnou hodnotu iteračního kroku.

První metoda využívající tohoto principu je založená na předpokladu, že kvalita výsledné cesty obchodního cestujícího závisí na průměrné vzdálenosti mezi městy. Z čehož plyne, že mapy obsahující více měst potřebují i výrazně více neuronů grafu. Proto se nejprve určí pro každé město jeho nejbližší sousední město a jejich vzdálenosti mezi sebou a pro tyto vzdálenosti se spočítá průměr $dist_{av}$. Následně se vytvoří čtverec o straně velikosti průměru $dist_{av}$. V tomto čtverci se musí vyskytovat předem definovaný počet neuronů. Poté se vydělí obsah volného prostoru čtvercem definovaným průměrem $dist_{av}$. Tento podíl určuje hodnotu, kterou se musí přenásobit definovaný počet neuronů na určený čtverec. Snižuje-li se průměrná vzdálenost $dist_{av}$, zvyšuje se počet neuronů a to tak, aby se mezi městy vyskytovalo podobné množství neuronů grafu.

Poslední navržený přístup předpokládá logaritmickou závislost počtu neuronů grafu na počtu měst. Ta je dána rovnicemi

$$m_s = m_{num} \cdot \frac{S_v}{S_d}, \tag{2.27}$$

$$p_c = 1 + \log_{1,7}(\mathrm{m_s}), \tag{2.28}$$

kde m_{num} je počet měst, S_v je obsah volného prostoru, S_d je obsah definovaného čtverce, m_s je parametr vyjadřující množství měst na S_d a p_c je parametr pro určení závislosti počtu neuronů na daný problém. Obsah S_d se určí ze vztahu

$$S_d = \left(\frac{w+h}{2}\right)^2 \cdot \frac{1}{10}, \tag{2.29}$$

kde w je šířka prostoru a h je výška prostoru. Pro obsah S_d byl experimentálně stanoven základ logaritmu na hodnotu $1,7$. Pokud by prostor obsahoval málo měst na definovaný obsah S_d, mohlo by se stát, že by hodnota parametru p_c mohla být příliš malá nebo dokonce záporná, což by mělo negativní dopad na kvalitu grafu. Proto pokud hodnota parametru p_c klesne pod hodnotu $0,75$, je parametr p_c nastaven na hodnotu $0,75$. Výsledný maximální počet neuronů n_{max} je dán vztahem

$$n_{max} = \frac{S_v}{S_d} \cdot p_m \cdot p_c, \tag{2.30}$$

kde p_m je parametr módu hustoty vrcholů. Nastavením tohoto parametru je ovlivňována samotná hustota vrcholů grafu. Nastavení parametru p_m je podrobněji rozebráno v podkapitole 3.2.

2.7 Problém algoritmu GNG

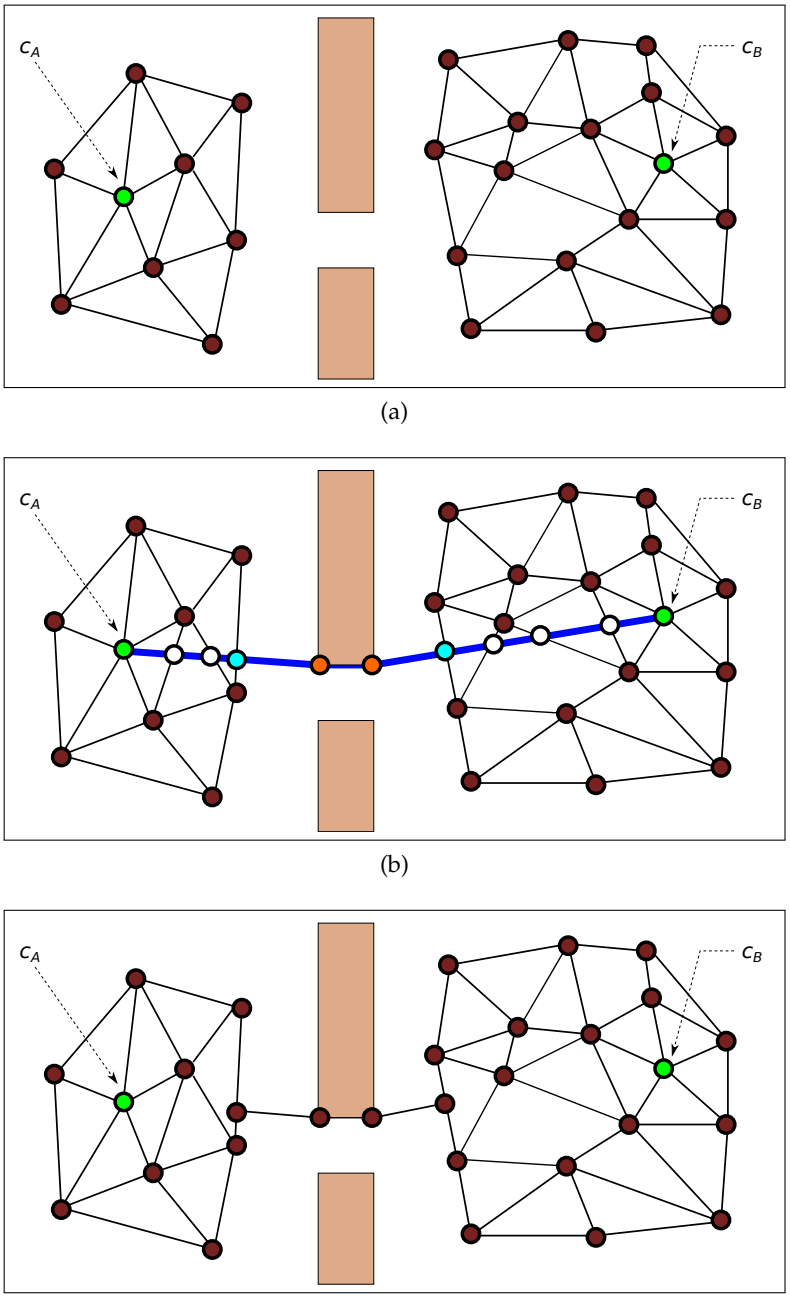
Primárním úkolem algoritmu GNG je popsat daná vstupní data, v našem případě to je volný prostor. Tuto úlohu algoritmus GNG zvládá, avšak v určitých zadaných prostředích se vyskytují specifikace, se kterými si samotný algoritmus nedovede poradit.

Problém nastává v prostředích, které obsahují úzké průchody mezi překážkami, jimiž by měla procházet cesta spojující města. Z důvodu těchto úzkých průchodů nemusí být nutně získán graf (reprezentovaný sítí neuronů) o jedné komponentě souvislosti, ale místo toho graf s více komponentami souvislosti. To je nepřípustné, protože nelze zajistit, aby algoritmus SOM pro TSP mohl najít cestu přes všechna města.

Z tohoto důvodu byl navržen následující algoritmus 4, který má z úkol propojit všechny komponenty souvislosti. Nejprve je nutné získat graf viditelnosti pro dané prostředí s městy. Dále se určí všechny komponenty souvislosti grafu, které obsahují alespoň jedno město. Ostatní komponenty souvislosti, které neobsahují města, budou smazány, protože jsou nepotřebné.

Pokud se počet komponent souvislosti nerovná jedné, vybere se první komponenta souvislosti a v ní se nalezne město (město A), jehož vzdálenost k jakémukoliv městu (město B) je minimální a které se nachází v libovolné jiné komponentě souvislosti. Město A a město B se spojí cestou, jejíž vrcholy tvoří cestu mezi městem A a B na grafu viditelnosti [2] . Tímto jsou spojeny obě komponenty souvislosti. Samozřejmě může nastat, že

[2]Graf viditelnosti je graf představující viditelnostní vztahy mezi jednotlivými vrcholy mapy.



Obrázek 2.4: Spojování komponent grafu, kde c_A a c_B jsou města, mezi kterými se vytvoří cesta na grafu viditelnosti (oranžové body, obrázek (b)). Tyrkysové body představují nejvzdálenější průsečíky od města dané komponenty s vytvořenou cestou a bílé body jsou ostatní průsečíky cesty s grafem. Na obrázku (c) je výsledné spojení dvou komonent souvislosti.

přidaná cesta do grafu bude protínat hrany či vrcholy již dříve vytvořeného grafu. Proto se nalezne průsečík získané cesty s komponentou A (komponenta souvislosti, v níž se nachází město A), který je nejdál od města A (zde je nutné, aby se vzdálenost počítala na grafu). Ostatní průsečíky s komponentou A a hrany vedoucí z nich budou smazány. To samé se zopakuje pro komponentu B, čímž se zajistí správné spojení komponent A i B.

Algoritmus 4: Algoritmus pro zajištění jedné komponenty souvislosti grafu

Vstup: $G(V, E, \epsilon)$ graf reprezentující mapu
Vstup: $G_v(V_v, E_v, \epsilon_v)$ graf viditelnosti
Výstup: $G(V, E, \epsilon)$ graf reprezentující mapu

```
1  begin
2      K ⟵ množina komponent souvislosti grafu G
3      k_num ⟵ |K|        // k_num určuje počet komponent souvislosti G
4      while k_num != 1 do
5          m_{k_1} ⟵ K_1
6          m_{k_i} ⟵ K_i
7          C cesta mezi m_{k_1} a m_{k_i} získaná z grafu G_v
8          G ⟵ G ∪ C    // přidání vrcholů a hran cesty C do grafu G
9          v_a ⟵ max{k_1 |v_{ik_1}, v_{k_j}|}
10         v_b ⟵ max{k_j |v_{ik_j}, v_{k_1}|}
11         C_new ⟵ cesta mezi v_a a v_b
12         G ⟵ G \ (C \ C_new)
13         K ⟵ znovu se určí množina komponent souvislosti grafu G
14         k_num ⟵ |K|
15 end
```

Následně se znovu získají všechny komponenty souvislosti grafu a opakuje se předchozí postup do té doby, dokud graf neobsahuje právě jednu komponentu souvislosti se všemi městy. Tímto je graf zcela připraven pro algoritmus SOM pro TSP.

Kapitola 3

Experimentální výsledky

Provedené experimenty jsou rozděleny na dvě části. V první části je studován vliv parametrů GNG na kvalitu řešení algoritmu SOM pro TSP. Druhá část experimentů porovnává vliv typu vstupního grafu na výsledek algoritmu SOM pro TSP. Porovnávány byly tři typy grafů a to graf viditelnosti, graf trojúhelníkové mřížky a graf GNG.

Kvality cest obchodního cestujícího jsou porovnávány podle dvou charakteristik. Těmi jsou výsledná délka cesty a čas potřebný k určení cesty. Délka cesty je určena následovně. Algoritmus samoorganizující se sítě nalezne pořadí měst, které reprezentuje cestu obchodního cestujícího. Délka cesty je určena součtem nejkratších možných vzdáleností mezi těmito městy.

Výsledné délky cest jsou porovnávány podle dvou charakteristik. První charakteristika vyjadřuje procentuální poměr mezi referencí ref (typicky se používá optimální vzdálenost cesty) a aritmetickým průměrem délek všech cest $\bar{l}$. Tato charakteristika se značí PDM a je určena vztahem

$$\mathrm{PDM} = \frac{\bar{l} - ref}{ref} \cdot 100\%. \tag{3.1}$$

Druhá charakteristika udává procentuální poměr mezi nejlepší délkou cesty l a referencí ref a označuje se jako PDB.

$$\mathrm{PDB} = \frac{l - ref}{ref} \cdot 100\%. \tag{3.2}$$

Všechny algoritmy jsou implementovány v jazyku C++ a zkompilovány kompilátorem G++ 4.2 s -02 optimalizací. Údaje z experimentů byly získány na počítačové sestavě s procesory Intel Xenon X5660 taktovaných na 2.8GHz, s pamětí 48GB RAM a s operačním systémem FreeBSD 8.2. Pro porovnávání kvality cest byly zvoleny dvě mapy a to potholes

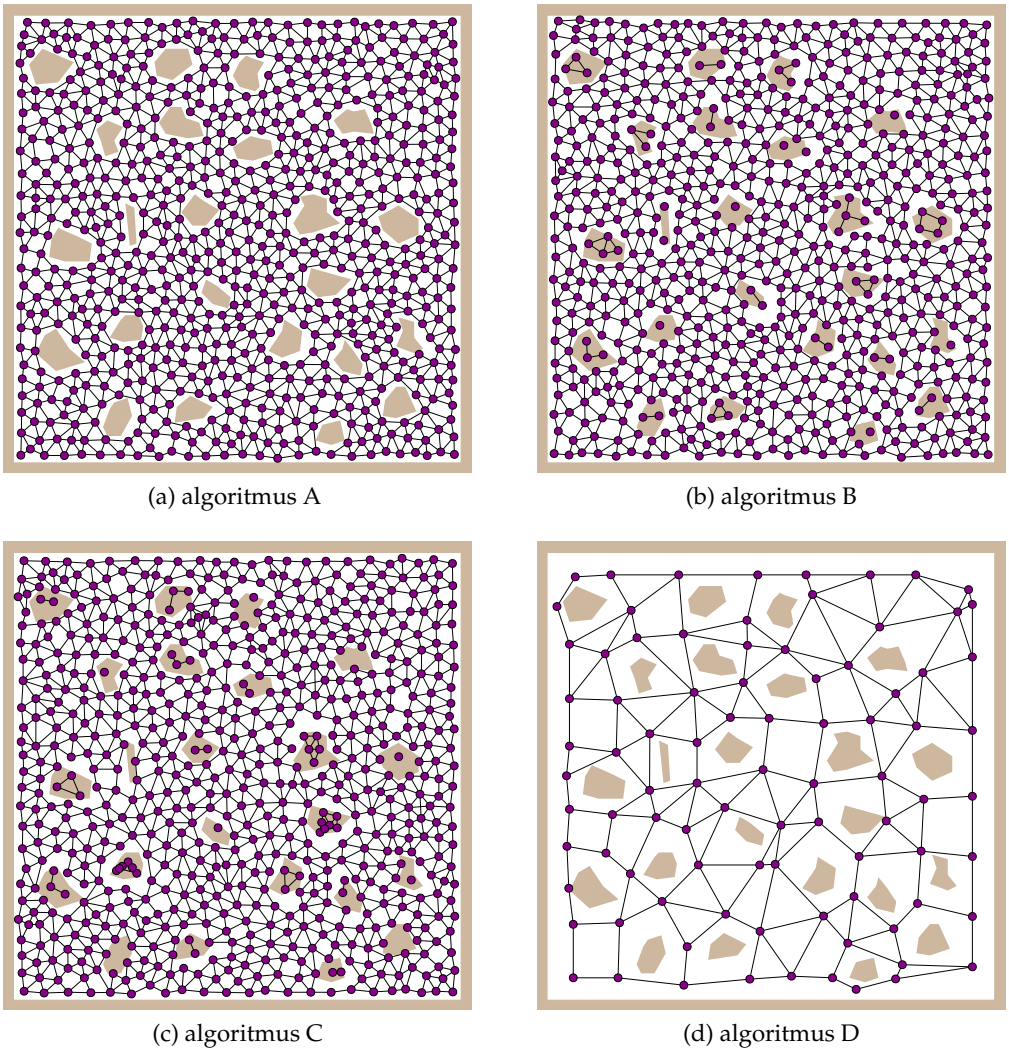
Tabulka 3.1: Počet měst pro jednotlivé typy map

	potholes	**potholes**$_1$	**potholes**$_2$	**jh-d200**	**jh-d400**
počet měst	17	282	68	180	89

a jh. Tyto mapy se dají ještě dělit na modifikace s odlišným počtem měst viz. tabulka 3.1.

3.1 GNG v polygonální doméně

V kapitole Neuronové sítě pro reprezentaci polygonální domény jsou v oddíle 2.6 popsány čtyři možné přístupy vypořádání se algoritmu GNG s překážkami. Je nutné zmínit, že tyto experimenty zatím nezkoumají vliv GNG na výsledky TSP, pouze se zde hledá vhodný přístup pro vytvoření grafu GNG, aby správně reprezentoval volný prostor. Požadovaná kritéria na graf jsou následující. Generovaný graf musí obsahovat co nejvíce požadovaných vrcholů, které zároveň musí být rovnoměrně rozprostřeny po celém volném prostoru.



(a) algoritmus A (b) algoritmus B

(c) algoritmus C (d) algoritmus D

Obrázek 3.1: Fialové body představují vrcholy (neurony) grafu GNG (v tomto případě je pro větší přehlednost vypnuto vymazávání nepotřebných neuronů). Pro tuto ukázku je zvolen maximální počet iterací tak, aby grafy měly stejné počáteční podmínky. Nejlepších výsledků dosáhl algoritmus A (a). U algoritmu B (b) je patrné, že hůře mapuje prostor okolo překážek. Algoritmus C (c) má obdobné výsledky jako algoritmus B, avšak uvnitř překážek se vyskytuje více nežádoucích vrcholů. Z obrázku (d) je patrné, že algoritmus D nedokázal vytvořit tolik vrcholů jako ostatní algoritmy.

Nejlepších výsledků dosahoval algoritmus A. Jeho graf rovnoměrně rozprostřel vr-

choly (neurony) po celém volném prostoru. Na obrázku 3.1a je vidět, že se v tomto případě nenachází žádný vrchol uvnitř překážky (všechny byly adaptovány mimo překážky) a že graf správně mapuje i okolí překážek.

V porovnání s algoritmem A algoritmus B 3.1b trochu zaostává. Především jeho zmapování okolí překážek není tak kvalitní. Je to dáno tím, že se nejprve neuronová síť rozprostře po celém prostředí, což znamená to, že se vrcholy rozmístí i na překážkážky a protože jsou takovéto vrcholy následně smazány, vznikají prázdná místa v okolí překážek a kvalita grafu se tak zhorší .

Algoritmus C na obrázku 3.1c se na první pohled výsledkem neliší od algoritmu B, avšak pokud algoritmus skončí a vrcholy nacházející se na překážkách se nesmažou, zjistí se, že některé vrcholy v překážce uvízly (tento jev se objevuje zejména u větších překážek). Je to dáno tím, že takový vrchol vznikl vně překážky a i přesto že mohl být vybrán jako nejbližší vrchol pro vstupní bod, nově adaptované souřadnice tohoto vrcholu se stále nacházely uvnitř překážky a vrchol se tak z ní nemohl dostat a zůstal uvnitř překážky. I když se graf s odfiltrovanými nepotřebnými vrcholy může kvalitativně měřit s algoritmem B (výsledky jsou prakticky identické), zůstává při vytváření grafu algoritmem C jistá nelogičnost.

Algoritmus D 3.1d by se mohl jevit jako teoreticky správný, ale v praxi moc dobře nefunguje. Je to způsobeno tím, že se graf postupně zvětšuje. Ze začátku je v grafu málo vrcholů, z čehož vyplývá, že se může lehce stát, že při vzniku spojení bude hrana reprezentující toto spojení křížit nějakou překážku, a proto spojení nevznikne. To může zabránit i vzniku nového vrcholu, čímž je narušena samotná podstata GNG.

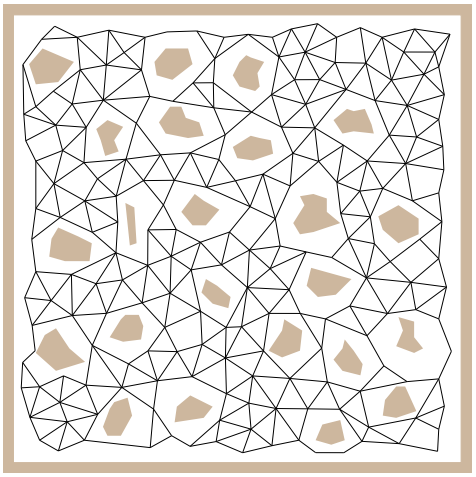
Nejlepších výsledků pro všechny mapy bylo dosaženo algoritmem A, který je nejbližší principu samotného algoritmu GNG. Sice se po skončení iterace musí zkontrolovat výsledná síť, zda se v ní nenachází nežádoucí vrcholy, avšak těch bývá minimum, typicky se i nevyskytují žádné. Z těchto důvodů se pro ukončování algoritmu GNG využíval algoritmus A.

3.2 Stanovení módů hustoty vrcholů pro generaci grafu GNG

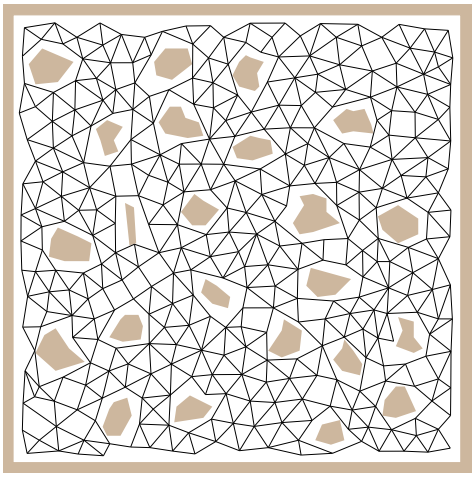
Protože nejvýznamnější vliv na kvalitu výsledků SOM pro TSP má hustota vrcholů grafu reprezentující volné prostředí, bylo stanoveno šest módů (very-low, low, medium, high, very-high a hyper) lišících se hustotou vrcholů grafu. Pro každý problém se napočítalo dvacet cest obchodního cestujícího, ze kterých se vyhodnocovaly výsledky experimentů.

Dynamické ukončování algoritmu GNG sice rozprostře po volném prostředí rovnoměrně vrcholy grafu, což je nesporná výhoda, ale má i jednu nevýhodu. Algoritmus GNG využívá prvku randomizace a není tak jisté, za jak dlouho se vrcholy do prostředí rozprostřou. Proto se může stát, že se bude čekat na jediný kontrolní bod, v jehož okolí se nevyskytuje žádný vrchol. Adaptování vrcholu do právě volného okolí kontrolního bodu může trvat určitou dobu a za tuto dobu se budou generovat další, již zbytečné vrcholy.

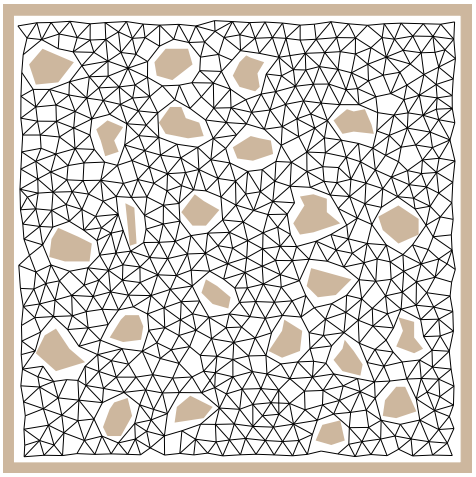
Tím se nejenom prodlužuje doba potřebná ke generaci grafu GNG, ale i doba nutná k spočítání cesty obchodního cestujícího. Tento jev je samozřejmě nežádoucí. Nicméně to nemění nic na faktu, že tento přístup zajišťuje rovnoměrné rozprostření vrcholů grafu generovaného algoritmem GNG, který obsahuje prvky randomizace, což přináší tomuto přístupu značné plus. Dále je nutno zmínit, že je zde prostor pro další zkoumání tohoto přístupu. Ten může být vylepšen tím, že pokud vymezený prostor obsahuje požadovaný



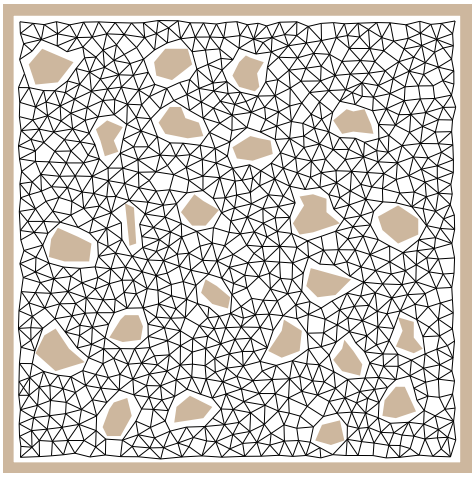
(a) very low



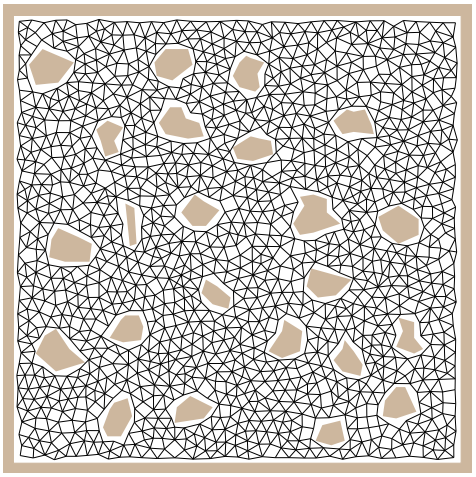
(b) low



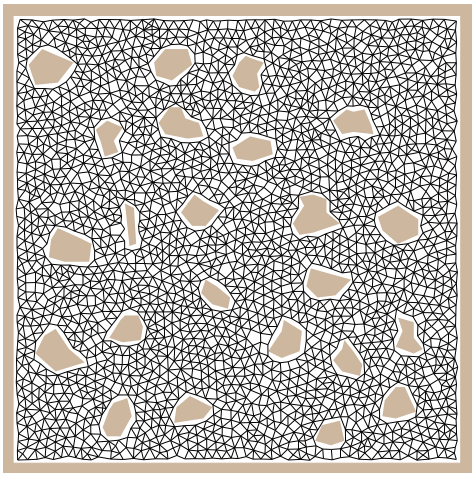
(c) medium



(d) high

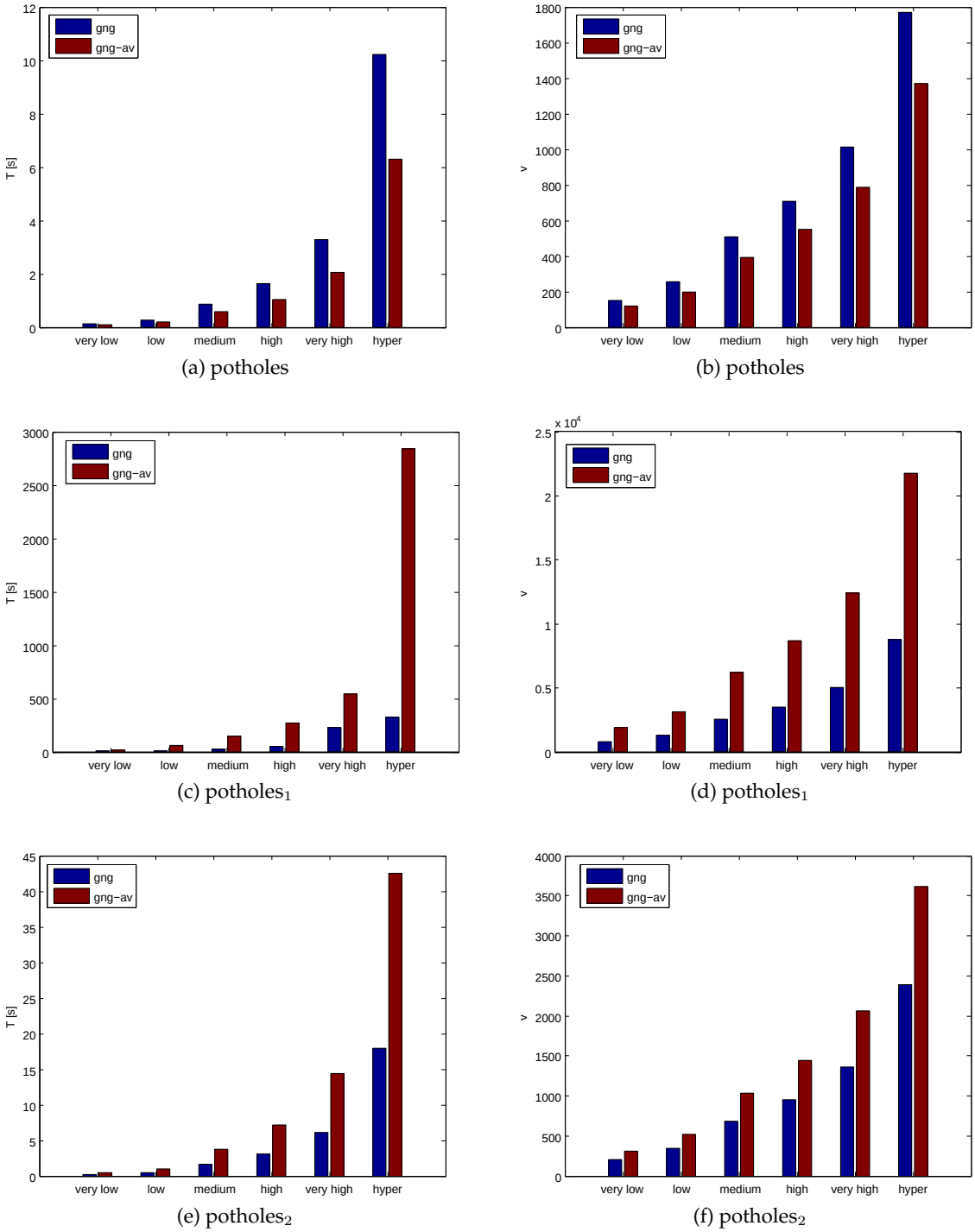


(e) very high



(f) hyper

Obrázek 3.2: Príklad zobrazení jednotlivých módů hustoty vrcholů na mapě potholes s vypnutou závislostí počnu neuronů na počtu měst



(a) potholes (b) potholes

(c) potholes$_1$ (d) potholes$_1$

(e) potholes$_2$ (f) potholes$_2$

Obrázek 3.3: Na obrázcích (a), (c), (e) jsou grafy závislosti počtu vrcholů V na módech hustoty. Grafy (b), (d), (f) jsou grafy zobrazující závislost mezi módy hustoty vrcholů a časem potřebným k vytvoření grafu T.

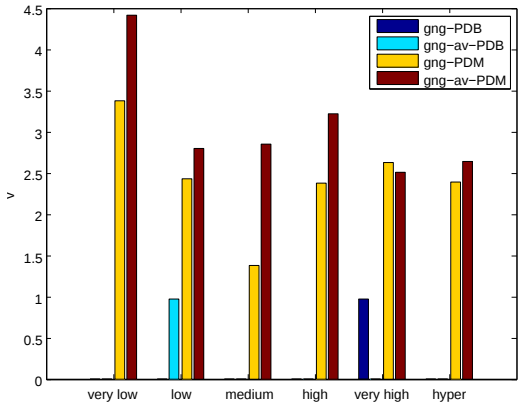
počet vrcholů, nebude s těmito vrcholy v tomto prostoru dále manipulováno a algoritmus bude adaptovat pouze ty vrcholy, pro něž toto pravidlo neplatí. Z důvodů časové nestability vytváření grafů nebyl tento přístup zakomponován do experimentů jakožto vstup al-

Tabulka 3.2: Parametry pro dynamické ukončování GNG

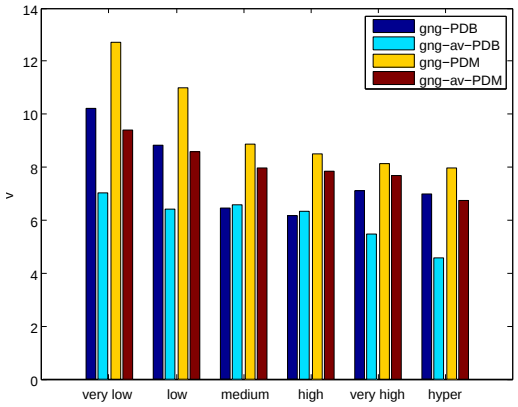
	very-low	**low**	**medium**	**high**	**very-high**	**hyper**
per [%]	7	5	4	3	2	1
$k_{kontrol}$	2000	10 000	25 000	40 000	70 000	100 000

goritmu SOM pro TSP. V tabulce 3.2jsou zobrazeny parametry potředmé k dynamickému ukončování algoritmu GNG. Parametr per je zde procentuální zlomek viz podkapitola 2.6 a parametr $k_{kontrol}$ je interval iteračních kroků, po jehož uběhnutí se zkontroluje, zda nemá být algoritmus GNG ukončen.

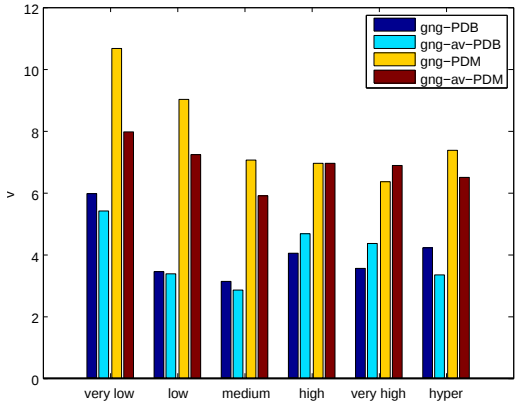
Ukončoval-li se algoritmus GNG po uplynutí určitého počtu iterací, využívaly se parametry z tabulky 3.3. Tento přístup se v praxi osvědčil, protože generoval grafy ve stabilním čase pro jednotlivé kvalitativní módy.



(a) potholes



(b) potholes$_1$



(c) potholes$_2$

Obrázek 3.4: Grafické vyjádření porovnání ukončovacích kritérií s předem daným maximálním počtem iteračních kroků. Gng-av představuje kritérium, při kterém se počítá s průměrnou vzdáleností mezi městy

Přístupy s pevnou hodnotou maximálního počtu iterací jsou rozděleny na dva typy

a to na přístup počítající s průměrnou vzdáleností mezi městy (v tabulkách značen jako gng-av) a na přístup předpokládající logaritmickou závislost mezi hustotou vrcholů a hustotou měst (v tabulkách značen jako gng). V tabulkách A.1 a A.2 jsou vypsány všechny parametry vygenerovaných grafů, některé parametry jsou vyobrazeny na grafech 3.3.

Ty ukazují, že přístup počítající s průměrnou vzdáleností mezi městy generuje daleko větší množství vrcholů (pro problémy s více městy) než přístup předpokládající logaritmickou závislost.

Tabulka 3.3: Parametry pro módy ukončování s max. počtem iterací.

	very-low	low	medium	high	very-high	hyper
$p_{kvality}$	3	5	10	14	20	35

Výsledky problému obchodního cestujícího na mapě potholes (s různým počtem měst) dokazují, že přístup počítající s průměrnou vzdáleností mezi městy vytváří až zbytečně mnoho vrcholů a hran grafu. Takovéto množství je zcela nepotřebné pro algoritmus SOM na grafu. Sloupcové grafy 3.4 ukazují, že oba přístupy jsou si poměrně rovnocenné, co se týče kvality cesty obchodního cestujícího, ale pokud započítáme do výsledného porovnání i dobu nutnou k určení cesty obchodního cestujícího a hlavně dobu nutnou k vytvoření grafu GNG, přístup předpokládající logaritmickou závislost zcela vítězí, proto se další testy prováděly pouze s tímto přístupem.

3.3 Problém obchodního cestujícího na grafu

V tomto oddíle jsou prezentovány výsledky porovnání kvalit řešení SOM pro TSP v závislosti na použitém typu grafu. Jako referenční algoritmus byl zvolen algoritmus short path aproximation, popsaný v [3].

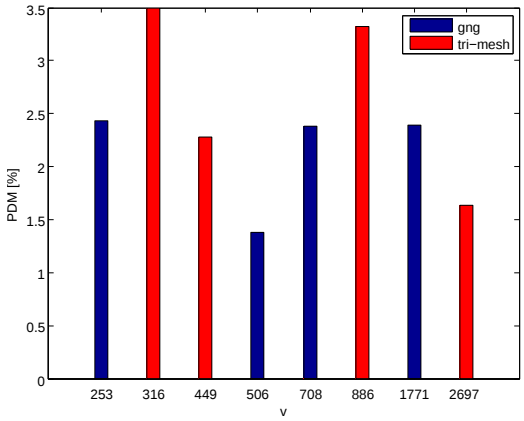
Kromě grafu GNG byla nalezena řešení úloh obchodního cestujícího také pro grafy trojúhelníkové mřížky a viditelnosti. Trojúhelníková mřížka byla získána generátorem `triangle` [12] poskytující mřížku, ve které trojúhelníky splňují podmínku maximálně zvoleného obsahu a minimálního úhlu. Pro každý problém bylo vygenerováno několik mřížek s různým počtem vrcholů.

Obsahuje-li prostředí méně měst (např. potholes), množství vrcholů grafu není tak důležité. Ze sloupcového grafu 3.5a je vidět, že je dokonce i obvyklé dosahování horších výsledků u grafů s větším počtem vrcholů než s nižším počtem. To je dáno randomizací algoritmu. V takovém případě je tedy vhodné použití grafů s nižším počtem vrcholů (z důvodů časové optimalizace).

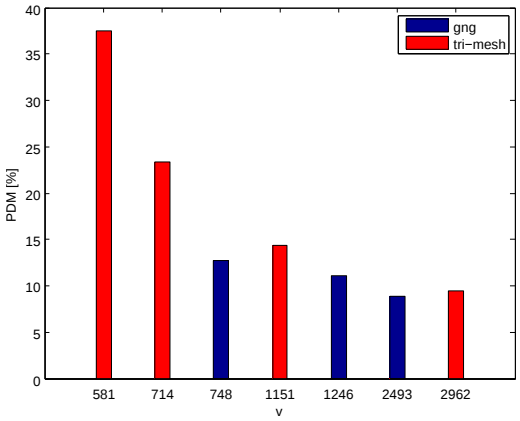
Pokud počet měst stoupne např. potholes$_2$ či jh-d400 (obě mapy mají podobné obsahy volného prostředí a obdobný počet měst, proto může být jejich hustota měst považována za porovnatelnou) je patrné, že graf trojúhelníkové mřížky s méně vrcholy ztrácí na graf GNG s obdobným počtem vrcholů. Ale při narůstajícím počtu vrcholů se tyto rozdíly ztrácí a výsledky pak lze považovat jako rovnocenné, co se kvality cesty týče. Mapy potholes$_1$ a jh-d200 obsahují poměrně hodně měst, ale i v jejich případech platí stejné závěry jako u map potholes$_2$ či jh-d400.

Při porovnání grafů GNG a trojúhelníkové mřížky s grafem viditelnosti jsou výsledky generované grafem viditelnosti přibližně o polovinu horší než výsledky s průměrným

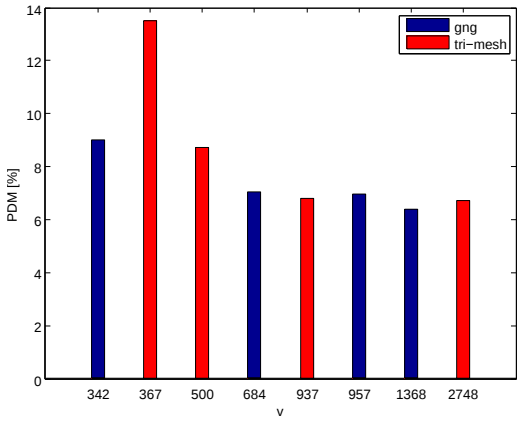
počtem vrcholů grafů rostoucího plynu či grafu trojúhelníkové mřížky. Porovnají-li se výsledky samoorganizující se sítě na grafu s přístupem založeném na aproximaci nejkratší cesty (spapprox), nedopadnou výsledky na grafu nejlépe. Pouze na mapě potholes (17 měst) se SOM na grafu vyrovná algoritmu využívajícího aproximaci nejkratší cesty.



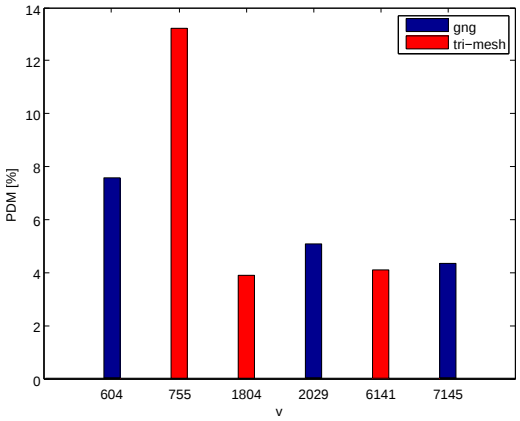
(a) potholes



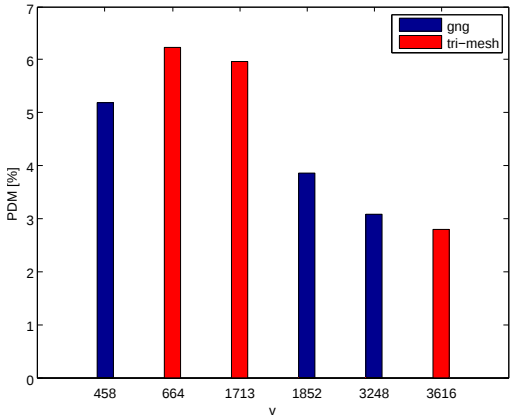
(b) potholes$_1$



(c) potholes$_2$



(d) jh-d200



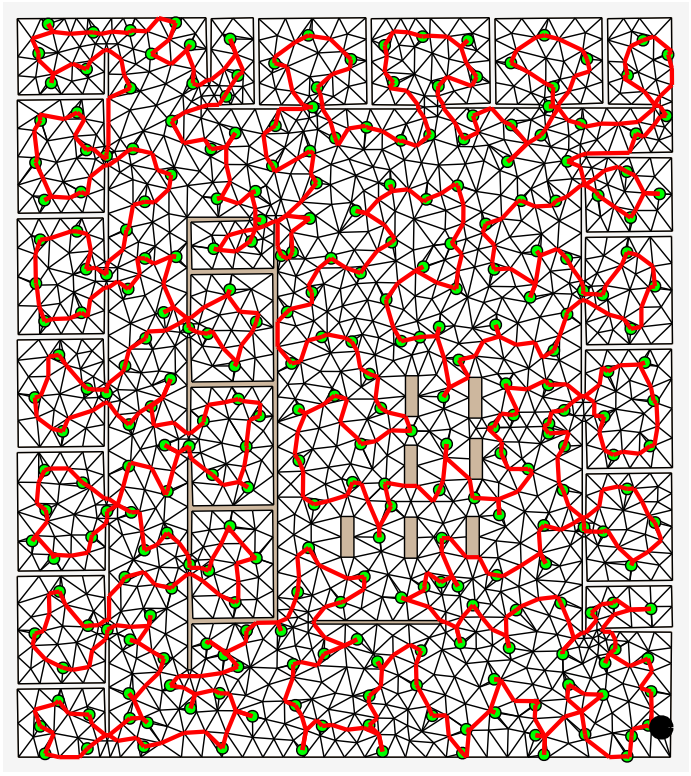
(e) jh-d400

Obrázek 3.5: Grafické vyjádření výsledků TSP na grafu rostoucího neuronového plynu a grafu trojúhelníkové mřížky.

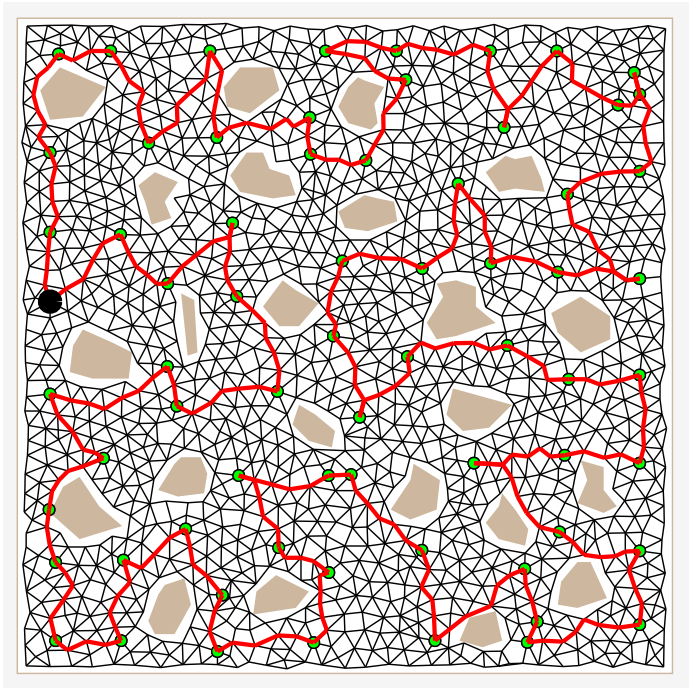
Studium vlivu typu grafu jakožto vstupu algoritmu SOM pro TSP jednoznačně ukázalo, že pro tento přístup je zcela nevhodný graf viditelnosti. Má-li se rozhodnout mezi grafem GNG a grafem trojúhelníkové mřížky, vychází jako vítěz graf trojúhelníkové mřížky. Přestože je nutné s tímto přístupem vygenerovat více vrcholů oproti grafu GNG, je čas potřebný ke generaci grafu kratší. Grafy trojúhelníkové mřížky se generovaly v řádech milisekund (i když byly generovány tisícovky vrcholů), zato grafy GNG se generovaly v řádech sekund. A toto je bohužel největší nevýhoda algoritmu GNG. Dále je nutno podotknout, že s narůstajícím se počtem vrcholů grafů se i ztrácí rozdíl mezi kvalitou výsledků TSP na grafu trojúhelníkové mřížky a na grafu GNG, což znovu zvýhodňuje prvně jmenovaný graf.

Tento závěr však graf algoritmu GNG nechce zatracovat, protože se vyskytují případy, kdy bude použití GNG výhodnější. Největší výhoda GNG se skrývá ve flexibilitě algoritmu. Algroritmus lze celkem snadno paralelizovat (a tím jej i zrychlit), což v dnešní době vícejádrových procesorů je velké plus. Dále je použití GNG výhodné, nastane-li například případ, kdy zpočátku nebude k dispozici úplná mapa volného prostředí. V takovém případě se může graf GNG utvářet i za právě probíhajícího prohledávání prostředí, což samozřejmě algoritmus generující trojúhelníkovou mřížku nedokáže.

Největší výhodu algoritmu GNG autor této práce vidí v jeho samotné podstatě, která vychází z biologických principů, díky nimž je algoritmus GNG flexibilní. A z biologických evolučních principů je známo, že dlouhodobě nepřežijí ti nejrychlejší a nejsilnější, ale ti co jsou schopni se co nejlépe přizpůsobit, a proto si algoritmy založené na neuronových sítích jako je GNG zaslouží další studium.



(a) jh-d200 trimesh



(b) potholes$_2$ gng very hight

Obrázek 3.6: Zobrazení výsledných cest pro problém obchodního cestujícího.

Kapitola 4

Závěr

Cílem této bakalářské práce je studium vlivu typu grafu na kvalitu řešení úlohy obchodního cestujícího (TSP) v polygonální doméně nalezené samoorganizující se sítí na grafu. Kromě grafů vycházejících z výpočetní geometrie (trojúhelníková mřížka a graf viditelnosti) byly studovány i přístupy založené na algoritmu neuronového plynu. Tento algoritmus je plně schopen poskytnout graf pro účely problému obchodního cestujícího, avšak jeho časová náročnost je příliš vysoká. Proto byl naimplementován i algoritmus rostoucího neuronového plynu (GNG), který vede k velmi podobným vlastnostem vygenerovaného grafu jako u neuronového plynu. Výhodou GNG je rychlejší konstrukce grafu než u algoritmu neuronového plynu, a proto byl pro další zkoumání používán pouze algoritmus GNG.

Hlavním problémem použití algoritmu GNG pro nalezení požadované reprezentace polygonální domény je podmínka na výsledný graf, který musí obsahovat pouze jednu komponentu souvislosti. Tento aspekt původní algoritmus GNG nezohledňuje, proto byl navržen nový přístup spojení případných více komponent souvislosti a to cestou získanou z grafu viditelnosti daného prostředí. Tím je zajištěno, že na výsledném grafu algoritmus pro obchodního cestujícího nalezne cestu přes všechna města.

Pro nalezení vhodného grafu, který by vedl na vyšší kvalitu řešení obchodního cestujícího algoritmem samoorganizující se sítě na grafu, byly nejdříve zkoumány metody zajišťující vygenerování grafu GNG o požadovaném počtu vrcholů. Bylo navrženo několik způsobů, jak požadovaného počtu vrcholů dosáhnout. První způsob, který je časově nejnáročnější, dynamicky ukončuje algoritmus GNG podle toho, jak jsou v grafu rozprostřeny vrcholy ve volném prostoru. Nevýhodou tohoto přístupu je, že ukončovací podmínka není dostatečně informovaná, neboť algoritmus pokračuje v přidávání vrcholů pro pokrytí volného prostoru i při dostačujícím množství vrcholů grafu. I přes časovou náročnost není tento přístup zavrhnutí hodný a je zde prostor pro další zkoumání a vylepšení, například adaptováním pouze těch neuronů, které se ještě správně nerozmístily do prostředí.

Další metody ukončování algoritmu GNG jsou navrženy tak, aby bylo předem jasné, v jakém iteračním kroku má algoritmus skončit. V první metodě je předpokládána závislost hustoty vrcholů grafu na průměrné délce mezi nejbližšími městy. Tato metoda se v praxi moc neosvědčila (kvůli generování zbytečně mnoha vrcholů grafu), ale z poznatků získaných implementací této metody vychází návrh poslední metody. Ta je založena na funkci, která vyjadřuje vztah mezi počtem měst a obsahem volného prostoru mapy. Tento vztah je vyjádřen logaritmickou závislostí mezi hustotou vrcholů grafu a hustotou měst v pro-

středí. Tím se dosáhlo obdobných výsledků TSP jako u první metody a to s podstatně nižším počtem vygenerovaných vrcholů grafu.

Z porovnání různých grafových reprezentací volného prostoro vyplývá, že jako nejvhodnější se jeví graf trojúhelníkové mřížky a to z důvodu obdobných výsledků cest obchodního cestujícího jako u grafu GNG a zároveň s výrazně nižšími výpočetními nároky na vytvoření grafu.

Avšak algoritmus GNG je velmi flexibilní, což mu přináší značné výhody. Lze jej dobře paralelizovat, čímž se generace grafu může urychlit. Dále lze graf vytvářet i tehdy, aniž bychom měli kompletní informace o prosředí, či je možno graf generovat pouze tam, kde je to požadováno. Proto pro určité specifické úlohy je využití algoritmu rostoucího neuronového plynu více než výhodné.

Literatura

[1] Angéniol, B.; de la C. Vaubois, G.; Texier., J.-Y. L.: Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, ročník 1, 1988: s. 289–293.

[2] Faigl, J.: *Multi-Goal Path Planning for Cooperative Sensing*. Dizertační práce, Czech Technical University in Prague, 2010.

[3] Faigl, J.; Kulich, M.; Vonásek, V.; aj.: An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. *Neurocomputing*, ročník 74, 2011: s. 671–679, doi:10.1016/j.neucom.2010.08.026.

[4] Faigl, J.; Mačák, J.: Multi-Goal Path Planning Using Self-Organizing Map with Navigation Functions. V *19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2011)*, 2011, s. 41–46.

[5] Faigl, J.; Přeučil, L.: Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals. V *ICANN 2011, Part I, LNCS 6791*, editace T. H. et al., Springer Verlag, 2011, s. 85–92.

[6] Fort, J. C.: Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biological Cybernetics*, ročník 59, č. 1, 1988: s. 33–40.

[7] Fritzke, B.: A Growing Neural Gas Network Learns Topologies. V *Advances in Neural Information Processing Systems 7*, MIT Press, 1995, s. 625–632.

[8] http://www.neuroinformatik.rub.de/VDM/research/gsn/JavaPaper/node16.html: Stav k 26.5.2011.

[9] Martinetz, T.; Schulten, K.: Topology representing networks. *Neural Networks*, ročník 7, č. 3, 1994: s. 507 – 522, ISSN 0893-6080.

[10] http://www.neuroinformatik.ruhr-uni-bochum.de/VDM/research/gsn/JavaPaper/node19.html: Stav k 26.5.2011.

[11] Preparata, F. P.; Shamos, M. I.: *Computational geometry: an introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, ISBN 0-387-96131-3.

[12] Shewchuk, J. R.: Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications*, ročník 22, 2001: s. 1–3.

[13] Somhom, S.; Modares, A.; Enkawa, T.: A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, 1997: s. 919–928.

[14] Somhom, S.; Modares, A.; Enkawa, T.: Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers and Operations Research*, ročník 26, č. 4, 1999: s. 395–407.

[15] Yamakawa, T.; Horio, K.; Hoshino, M.: Self-Organizing Map with Input Data Represented as Graph. V *ICONIP (1)*, 2006, s. 907–914.

Dodatek A

Tabulky s výsledky experimentů

Tabulka A.1: Ukončovacím kritériem počítajícím s logaritmickou závislostí (gng)

mapa		vrcholů	hran	kroků	T [s]
potholes					
	very-low	151	692	99 660	0,127
	low	253	1218	166 980	0,264
	medium	506	2528	333 960	0,871
	high	708	3602	467 280	1,627
	very-high	1012	5240	667 920	3,289
	hyper	1771	9400	$1{,}168 \cdot 10^6$	10,235
potholes$_1$					
	very-low	748	3774	49 3680	2,693
	low	1246	6542	82 2360	6,598
	medium	2493	13 348	$1{,}645 \cdot 10^6$	23,986
	high	3491	18 956	$2{,}304 \cdot 10^6$	48,221
	very-high	4987	27 254	$3{,}291 \cdot 10^6$	228,030
	hyper	8725	47 992	$5{,}760 \cdot 10^6$	323,720
potholes$_2$					
	very-low	205	929	13 5300	0,242
	low	342	1670	22 5720	0,508
	medium	684	3467	45 1440	1,607
	high	957	4956	63 2280	3,031
	very-high	1368	7182	90 2880	6,071
	hyper	2395	12 762	$1{,}580 \cdot 10^6$	17,985

Tabulka A.2: Ukončovací kritérium počítajícím s prům. vzdáleností měst (gng-av)

mapa		vrcholů	hran	kroků	T [s]
potholes					
	very-low	116	504	77 220	0,092
	low	196	928	129 360	0,185
	medium	392	1918	258 720	0,564
	high	549	2752	362 340	1,044
	very-high	785	4034	518 100	2,067
	hyper	1373	7176	906 180	6,305
potholes$_1$					
	very-low	1863	9924	$1{,}229{\cdot}10^6$	14,575
	low	3106	16 798	$2{,}049{\cdot}10^6$	55,254
	medium	6213	33 986	$4{,}100{\cdot}10^6$	147,081
	high	8696	47 890	$5{,}740{\cdot}10^6$	266,562
	very-high	12 423	68 828	$8{,}201{\cdot}10^6$	536,180
	hyper	21 736	121 142	$1{,}435{\cdot}10^6$	2834,840
potholes$_2$					
	very-low	310	1494	204 600	0,464
	low	517	2600	341 220	1,051
	medium	1034	5352	683 100	3,748
	high	1449	7612	956 340	7,112
	very-high	2069	11 016	$1{,}366{\cdot}10^6$	14,353
	hyper	3623	19 502	$2{,}391{\cdot}10^6$	42,580

Tabulka A.3: Potholes - ukončovací kritérium s maximálním počtem iteračních kroků.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-av-very-low	116	504	4,41	0,00	3	26
gng-av-low	196	928	2,79	0,97	4	28
gng-av-medium	392	1918	2,85	0,00	6	32
gng-av-high	549	2752	3,22	0,00	7	35
gng-av-very-high	785	4034	2,51	0,00	10	38
gng-av-hyper	1373	7176	2,64	0,00	17	44
gngr-very-low	151	692	3,38	0,00	4	26
gng-low	253	1218	2,43	0,00	5	29
gng-medium	506	2528	1,38	0,00	8	34
gng-high	708	3602	2,38	0,00	10	36
gng-very-high	1012	5240	2,63	0,97	12	42
gng-hyper	1771	9400	2,39	0,00	19	47
spapprox	–	–	1,63	0,00	42	32

Tabulka A.4: Potholes_1 - ukončovací kritérium s maximálním počtem iteračních kroků.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-av-very-low	1863	9924	9,39	7,01	187	15,272
gng-av-low	3106	16 798	8,56	6,42	257	25,283
gng-av-medium	6213	33 986	7,98	6,56	443	23,787
gng-av-high	8696	47 890	7,82	6,31	581	26,533
gng-av-very-high	12423	68 828	7,68	5,46	808	30,081
gng-av-hyper	21736	121 142	6,75	4,55	1407	37,273
gng-very-low	748	3774	12,72	10,22	132	11,923
gng-low	1246	6542	11,01	8,82	155	13,620
gng-medium	2493	13 348	8,88	6,43	223	24,005
gng-high	3491	18 956	8,48	6,16	277	19,493
gng-very-high	4987	27 254	8,13	7,09	362	21,135
gng-hyper	8725	47 992	7,96	6,96	635	26,839
spapprox	–	–	7,22	5,39	354	5,177

Tabulka A.5: Potholes$_2$ - ukončovací kritérium s maximálním počtem iteračních kroků.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-av-very-low	310	1494	7,96	5,40	14	539
gng-av-low	517	2600	7,22	3,36	16	596
gng-av-medium	1034	5352	5,88	2,82	26	723
gng-av-high	1449	7612	6,94	4,65	32	791
gng-av-very-high	2069	11 016	6,87	4,33	42	969
gng-av-hyper	3623	19 502	6,46	3,33	73	1.378
gng-very-low	205	930	10,65	5,93	12	498
gng-low	342	1670	9,00	3,41	14	553
gng-medium	684	3466	7,02	3,09	19	973
gng-high	957	4956	6,94	4,03	33	813
gng-very-high	1368	7182	6,35	3,51	31	803
gng-hyper	2395	12 762	7,36	4,21	48	944
spapprox	–	–	5,65	2,84	51	293

Tabulka A.6: Potholes - porovnání PDM pro GNG a grafy trojúhelníkové mřížky.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-very-low	151	692	3,38	0,00	4	26
gng-low	253	1218	2,43	0,00	5	29
gng-medium	506	2528	1,38	0,00	8	34
gng-high	708	3602	2,38	0,00	10	36
gng-very-high	1012	5240	2,63	0,97	12	42
gng-hyper	1771	9400	2,39	0,00	19	47
trimesh299	316	829	3,50	0,00	6	30
trimesh432	449	1206	2,28	0,79	9	57
trimesh869	886	2448	3,32	0,00	9	38
trimesh2680	2697	7714	1,63	0,00	25	55
vis	170	6164	7,50	0,37	10	21
spapprox	–	–	1,63	0,00	42	32

Tabulka A.7: Potholes$_1$ - porovnání PDM pro GNG a grafy trojúhelníkové mřížky.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-very-low	748	3774	12,72	10,22	132	11,923
gng-low	1246	6542	11,01	8,82	155	13,620
gng-medium	2493	13 348	8,88	6,43	223	24,005
gng-high	3491	18 956	8,48	6,16	277	19,493
gng-very-high	4987	27 254	8,13	7,09	362	21,135
gng-hyper	8725	47 992	7,96	6,96	635	26,839
trimesh299	581	1624	37,54	31,05	169	10,124
trimesh432	714	2001	23,37	17,64	132	10,489
trimesh869	1151	3243	14,28	11,36	152	12,075
trimesh2680	2962	8509	9,45	7,70	242	16,575
vis	435	52468	19,22	14,20	255	7,211
spapprox	–	–	7,22	5,39	354	5,177

Tabulka A.8: Potholes$_2$ - porovnání PDM pro GNG a grafy trojúhelníkové mřížky.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-very-low	205	930	10,65	5,93	12	498
gng-low	342	1670	9,00	3,41	14	553
gng-medium	684	3466	7,02	3,09	19	973
gng-high	957	4956	6,94	4,03	33	813
gng-very-high	1368	7182	6,35	3,51	31	803
gng-hyper	2395	12 762	7,36	4,21	48	944
trimesh299	367	982	13,49	9,68	13	677
trimesh432	500	1359	8,68	4,80	16	585
trimesh869	937	2601	6,78	4,18	22	676
trimesh2680	2748	7867	6,69	4,37	55	888
vis	221	11594	14,65	11,23	36	605
spapprox	–	–	5,65	2,84	51	293

Tabulka A.9: Mapa jh-d200 - porovnání PDM pro GNG a grafy trojúhelníkové mřížky.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-very-low	604	2748	7,57	5,73	45	8,503
gng-low	1012	4818	5,88	3,13	56	5,841
gng-medium	2029	10 272	5,07	3,84	91	11,239
gng-high	2851	14 682	4,87	3,33	124	7,893
gng-very-high	4075	21 430	4,75	2,59	167	8,071
gng-hyper	7145	38 276	4,33	2,79	289	14,459
trimesh575	755	1993	13,21	10,18	51	7,514
trimesh1624	1804	4886	5,96	3,91	82 s	5,775
trimesh3527	3707	10 249	4,26	2,36	148	7,687
trimesh5961	6141	17 354	4,09	3,11	232	10,602
vis	376	18 652	14,83	10,80	68	4,367
spapprox	–	–	3,39	2,17	130	2,596

Tabulka A.10: Mapa jh-d400 - porovnání PDM pro GNG a grafy trojúhelníkové mřížky.

Graf	**Vrcholů**	**Hran**	**PDM** [%]	**PDB** [%]	$\mathbf{T_i}$ [ms]	$\mathbf{T_s}$ [ms]
gng-very-low	278	1150	6,19	3,93	15	905
gng-low	458	2012	5,19	3,13	17	984
gng-medium	923	4358	4,83	3,37	26	1180
gng-high	1296	6388	4,32	2,72	32	1268
gng-very-high	1852	9312	3,86	2,54	44	1577
gng-hyper	3248	16 878	3,09	1,69	75	1748
trimesh575	664	1720	6.23	3,39	21	1058
trimesh1624	1713	4613	3,87	1,97	40	1343
trimesh3527	3616	9976	2,80	1,38	76	1708
trimesh5961	6050	17 081	2,76	1,47	125	2019
vis	285	9388	9,94	6,51	25	643
spapprox	–	–	2,09	0,75	119	869

Dodatek B

Obsah CD

Přiložené CD obsahuje zdrojové kódy pro programy gng a tsp-somhom text diplomové práce ve formátu PDF a zdrojové kódy celého textu pro systém LaTeX. V následující tabulce je popsána struktura CD.

Tabulka B.1: Adresářová struktura na CD

Adresář	Popis
`programming\programs`	zdrojové kódy programů
`programming\src`	zdrojové kódy knihovny
`thesis\src`	zdrojové kódy textu diplomové práce
`thesis\thesis.pdf`	text diplomové práce