Možnosti využitia ľudského postupu pre návrh algoritmov na riešenie ťažkých problémov

Diplomová práca

Matej Lučenič

Univerzita komenského v bratislave

Fakulta matematiky, fyziky a informatiky

Katedra informatiky

Informatika

Doc. RNDr. Juraj Procházka Csc.

Bratislava 2009

Abstrakt

Lučenič Matej: *Možnosti využitia ľudského postupu pre návrh algoritmov na riešenie ťažkých problémov* [Diplomová práca] – Komenského Univerzita, Bratislava, Fakulta Matematiky, Fyziky a Informatiky; Katedra Informatiky. – Školiteľ: doc. RNDr. Juraj Procházka, CSc. Bratislava : Komenského Univerzita 2009. 42 strán.

Experimentálne zisťované parametre ľudského riešenia vizuálne zadaného problému obchodného cestujúceho ukazujú, že človek vie tento problém riešiť veľmi efektívne, pričom dosahuje pomerne kvalitné výsledky. Otázkou je, či je možné na základe pozorovania ľudského postupu navrhnúť algoritmus, ktorý by dokázal riešiť problém s podobným úspechom. Hlavnou výhodou človeka je globálny pohľad na vizuálne prezentovaný problém, ktorý počítač dokáže vyvážiť rýchlosťou svojej práce a vyššou schopnosťou rozlíšenia drobných rozdielov vo vzdialenostiach vrcholov. V tejto práci na základe výsledkov pozorovania ľudského postupu navrhujeme algoritmus pre riešenie problému obchodného cestujúceho v rovine a analyzujeme jeho vlastnosti v porovnaní s algoritmami podobného typu. Výsledkom je algoritmus lineárneho prehľadávania, ktorý pracuje v čase $O(n^4)$ a kvalita jeho riešení je porovnateľná a v niektorých prípadom lepšia, ako kvalita riešení porovnávaných algoritmov.

Kľúčové slová: *problém obchodného cestujúceho – hamiltonovská kružnica – lineárne prehľadávanie – konvexný obal – metóda najlacnejšieho vkladania – lokálna optimalizácia*

Poďakovanie

Chcel by som sa na tomto mieste poďakovať všetkým, ktorí prispeli k tomu, že sa mi podarilo dokončiť túto prácu. V prvom rade môjmu školiteľovi, doc., RNDr. Jurajovi Procházkovi, CSc., za trpezlivosť, podporu a rady, ktoré mi pomohli pri práci, Mgr. Zuzane Macovej za trpezlivosť a korektúry záverečného textu, kolegom z práce za technologickú podporu, a tak isto mojim nadriadeným, ktorí mi umožnili venovať sa tejto práci popri pracovných povinnostiach. Špeciálne by som chcel poďakovať Lucii Ďubekovej za trpezlivosť a podporu počas celej záverečnej časti môjho štúdia.

Obsah

1. Úvod

Počítač vždy slúžil človeku ako pomocník na „ťažkú“ prácu. Oproti človeku má viacero veľkých výhod. Medzi jeho hlavné prednosti patrí neúnavnosť a schopnosť pracovať veľmi rýchlo. Napriek týmto výhodám sa informatika po čase dostala k množine praktických problémov, pre ktoré neexistuje algoritmus, ktorý by umožňoval počítaču problém vyriešiť v reálnom čase, prípadne s reálnymi zdrojmi. Zložitosť niektorých problémov jednoducho presahuje schopnosti súčasných počítačov.

Pri riešení takýchto problémov sa využívajú tri základné prístupy. Prvý spôsob je obmedzenie množiny vstupov na takú, pre ktorú sa podarí nájsť algoritmus, ktorý je schopný na tejto obmedzenej množine pracovať v porovnaní s dovtedy využívanými algoritmami oveľa efektívnejšie. Takéto algoritmy vedú k polynomiálnym časom pre riešenie problému.

Druhý prístup ponecháva kompletnú množinu vstupov problému a povoľuje obmedzenia na kvalitu riešenia problému. Buď je dosiahnutý výsledok poznačený určitou konkrétnou nepresnosťou, alebo je výsledok správny s určitou pravdepodobnosťou. Takéto výsledky majú veľký význam pre praktickú stránku problémov, lebo poskytujú, aj keď nie absolútne presné, ale vo veľkej miere postačujúce riešenie problému. Tak isto je možné podľa nich stanoviť obmedzenie, na hraničné hodnoty optimálnych riešení, ktoré umožňujú následné zefektívnenie práce deterministických metód. Sem patria napríklad aj genetické algoritmy, simulované žíhanie. Všetky sú založené na lokálnom prehľadávaní množiny riešení s určitým aspektom znáhodnenia prehľadávania..

Tretí prístup, ktorý je vlastne z historického hľadiska prvým v poradí, je snaha o absolútne riešenie problému. Tento prístup pri ťažkých problémoch spočíva väčšinou v implementácií kompletného prehľadávania množiny riešení daného problému. V zásade podľa mojich (do veľkej miery obmedzených) informácií v poslednom čase nedošlo v tejto oblasti ku prelomovému objavu, alebo riešeniu. Deterministické metódy vedú k exponenciálnej alebo dokonca faktoriálenej zložitosti. A celý informatický svet stojí pred otázkou P=NP?.

Algoritmus je pre počítač postup na vykonanie práce. V podstate vie vykonať len to, čo mu človek prikáže pomocou príkazov programovacieho jazyka. Aj algoritmy, ktoré sa dokážu učiť na základe dopredu pripravenej množiny zadaní a riešení, prípadne počas učebného procesu vedeného človekom, v skutočnosti pracujú len s informáciami, ktoré do nich niekto fyzicky vložil. Prípadne sa počíta s tým, že „väčšie“ inštancie problému sú v zásade podobné menším, takže keď „vychováme“ algoritmus na menších zadaniach, bude si vedieť na väčších už poradiť sám. Tento proces učenia má byť podobný procesu učenia ľudského mozgu, o ktorom existuje veľký objem teórie aj praktických štúdií. Majú však jednu hlavnú spoločnú myšlienku. A tou je, že aj ľudský mozog dokáže pracovať len s informáciou, ktorá sa do neho nejakým spôsobom dostane. Veľkou neznámou stále zostávajú objavy, prípadne teórie, ktoré sa úplne vymykajú aktuálnemu spôsobu myslenia a úrovni vedomostí. Tu sa veda uberá smerom hľadania „geniálneho“ mozgu a odpovede hľadá napríklad v spôsoboch jeho využívania. Bežný človek podľa súčasných poznatkov využíva svoj mozog len na 10 percent. Otázkou je, či spôsob využívania mozgu, prípadne využívanie jeho väčšej časti je naozaj to, čo umožňuje výnimočným ľuďom posúvať vývoj ľudskej spoločnosti dopredu.

Pozrime sa teraz na spôsob riešenia problémov pomocou počítača z iného pohľadu. Jednoduchým príkladom môže byť riešenie kvadratickej rovnice. Predstavme si, že nepoznáme Vietove vzorce na riešenie kvadratickej rovnice. Ako by sme takýto problém riešili na počítači? V podstate hneď sa ponúka nasledovný jednoduchý postup:

- Zostrojíme algoritmus, ktorý jednoduchým spôsobom zobrazí graf kvadratickej funkcie.
- Z grafu budeme vidieť, na koľkých miestach pretne graf funkcie os x.
- Podľa toho určíme počiatočné body intervalov pre počítanie koreňov
- Napíšeme algoritmus, ktorý napríklad metódou polenia intervalov vypočíta korene rovnice (v prípade dvoch koreňov) alebo postupným výpočtom menších a menších hodnôt dospeje ku koreňu (v prípade dvojnásobného koreňa).

Čo sme vlastne urobili? Bez hlbšej znalosti problému sme využili schopnosť počítača neúnavne pracovať na jeho vyriešení. Ak by sme však mali hlbšie vedomosti o probléme, reprezentované v tomto prípade znalosťou Vietových vzorcov pre výpočet koreňov

kvadratickej rovnice, náš algoritmus by pracoval oveľa rýchlejšie. Ak vezmeme do úvahy, že výpočet hodnôt trvá v oboch prípadoch rovnako dlho (konštantný čas), algoritmus, využívajúci hlbšiu vedomosť o probléme, pracuje v konštantnom čase.

Tu je naznačená jedna z ciest ku hlavnej myšlienke tejto práce. Podstatná pri riešení akéhokoľvek problému pomocou počítača je úroveň vedomostí o probléme toho, kto algoritmus vytvára, nie hrubá pracovná sila počítača, ani jeho rýchlosť. Všetko, čo počítač robí alebo nerobí, zvládne alebo nezvládne, závisí len od toho, kto preňho chystá návod na prácu. To znamená, že podstatný v tomto smere je mozog programátora a jeho spôsob uvažovania a miera jeho poznania problému.

Toto tvrdenie môžeme zovšeobecniť na človeka. Keďže veľké množstvo problémov v informatike má jednoduché, praktické zadanie, ktoré nie je veľmi vzdialené chápaniu bežných ľudí, ponúka sa ako možnosť riešenia problému s pomocou širšieho okruhu ľudí. Nakoniec, hlavné pokroky klasickej vedy, napríklad v oblasti fyziky, v dávnejších dobách vychádzali do veľkej miery z experimentálnych výsledkov a pozorovaní prírody, na základe ktorých boli následne formulované teórie. Po určitom čase sa schopnosti pozorovať prírodu rozšírili, napríklad o výkonnejšie prístroje, ktoré umožnili hlbší pohľad na veci okolo nás a teória sa znovu o kúsok viac priblížila pozorovanej realite.

Ako metóda pre návrh algoritmu sa podľa predošlých úvah ponúka nasledovné:

Keďže pre riešenie problémov je tak, či onak podstatný a hlavný aktér človek s jeho schopnosťou využívať vlastný mozog, je možno potrebné sa sústrediť na neho a počítač mu ponúknuť ako prostriedok pre zber informácií, ktoré mu môžu pomôcť vyriešiť problém, miesto spoliehania sa na počítač, že vyrieši problém zaňho. Človek môže byť a aj vždy bude pre počítač ten geniálny vodca, ktorý mu umožní hlbší náhľad na problém a v konečnom štádiu aj jeho vyriešenie.

Sú dve možnosti, ako takúto spoluprácu využiť:

- Algoritmus, ktorý má riešiť problém, bude obsahovať rozhranie pre spoluprácu s „vyšším vedením“, ktoré bude predstavovať človek. Počítač ponúkne človeku vždy

medzivýsledky svojej práce, a hodnotenie bude ne človeku, ktorý určí smer ďalšieho výpočtu.

- Vytvoríme algoritmus umožňujúci v širšom meradle zbierať informácie o riešení probléme od ľudí, ktorým umožní prostredníctvom jednoduchého rozhrania, prípadne aj výsledkov svojich výpočtov, problém riešiť. Z následného vyhodnotenia riešenia problému bude možné formulovať teóriu, ktorú sa pokúsi experimentátor dokázať. Výsledkom nakoniec bude algoritmus, ktorého schopnosti riešiť problém budú ďalej analyzované.

Použijeme experimentálny postup druhého typu. Postup bude nasledovný:

- Výber problému,
- Analýza doteraz dosiahnutých výsledkov a poznatkov o riešení tohto problému za pomoci človeka,
- Naprogramovanie rozhrania pre riešenie problému človekom spolu s algoritmami, ktoré dodajú človeku približný pohľad na optimálne riešenie,
- Experiment a analýza výsledkov,
- Na základe výsledkov experimentov pokus o návrh algoritmu pre riešenie problému,
- Odhad zložitosti algoritmu,
- Diskusia o dosiahnutých výsledkoch a zhodnotenie experimentu.

Samozrejme, postup by mal v sebe obsahovať aj dôkaz správnosti a zložitosti algoritmu.

2. Výber problému

Dôležitou časťou experimentu je výber problému. Musí byť vhodný na prezentáciu riešiteľom a mať dostatočne jednoduché zadanie. Samozrejme veľkou výhodou je aj spojenie s nejakým praktickým problémom z reálneho života a možnosť vhodnej grafickej vizuálnej prezentácie, ktorá by umožňovala jednoduché prístup pre riešiteľa. Vzhľadom na možnosť prevedenia jedného problému na iný (NP úplnosť) je vhodné vybrať problém, ktorého riešenie bude možné využiť aj pre riešenie ďalších z množiny ťažkých problémov.

V rámci tejto práce bol zvolený problém, ktorý má veľmi dlhú históriu a spĺňa všetky hore uvedené požiadavky. Ide o takzvaný Problém obchodného cestujúceho. Pre potreby jednoduchej grafickej reprezentácie som zvolil nasledovné zadanie:

Je daný súbor vrcholov (bodov na obrazovke), so známymi súradnicami. Úlohou je nájsť najkratšiu cestu, spájajúcu všetky vrcholy, ktorá prechádza každým vrcholom práve raz a začína a končí v jednom vrchole.

Toto je „ľudová“ formulácia problému, ktorá je po uvedení príkladu väčšine ľudí jasná a zrozumiteľná. V jazyku teórie grafov ide o nájdenie najkratšej hamiltonovskej kružnice v rovinnom neorientovanom grafe o n vrcholoch, takzvaný Euklidovský problém obchodného cestujúceho pre dvojrozmerný prípad. Tento problém vo všeobecnosti patrí do množiny NP úplných problémov [6].

Pre riešenie problému existuje naozaj veľmi pestrá paleta algoritmov zo všetkým možných oblastí prístupu. Avšak žiadny deterministický algoritmus, ktorý by vedel nájsť optimálne riešenie v reálnom čase, zatiaľ neexistuje. Niektoré zo známych algoritmov je možné použiť pre porovnanie riešení človeka, respektíve navrhovaného algoritmického riešenia a bežne používaných metód podobného typu v tejto oblasti.

3. Analýza doteraz dosiahnutých výsledkov a poznatkov v oblasti ľudského riešenia TSP

Na podobnú tému bolo vykonaných niekoľko experimentov a publikovaných viacero článkov. Okrem informatiky táto téma zasahuje aj do oblasti psychológie, takže niektoré publikácie sa témou zaoberajú aj z tohto pohľadu. Prístupy k riešeniu problému by sme mohli rozdeliť nasledovne:

- Riešenie problému človekom.
- Riešenie problému počítačom, pričom v určitom okamihu človek vstupuje do výpočtu.

V nasledujúcej časti uvedieme stručný historický prehľad väčšiny verejne publikovaných experimentov na tému ľudského riešenia TSP zoradených podľa roku, v ktorom boli publikované výsledky:

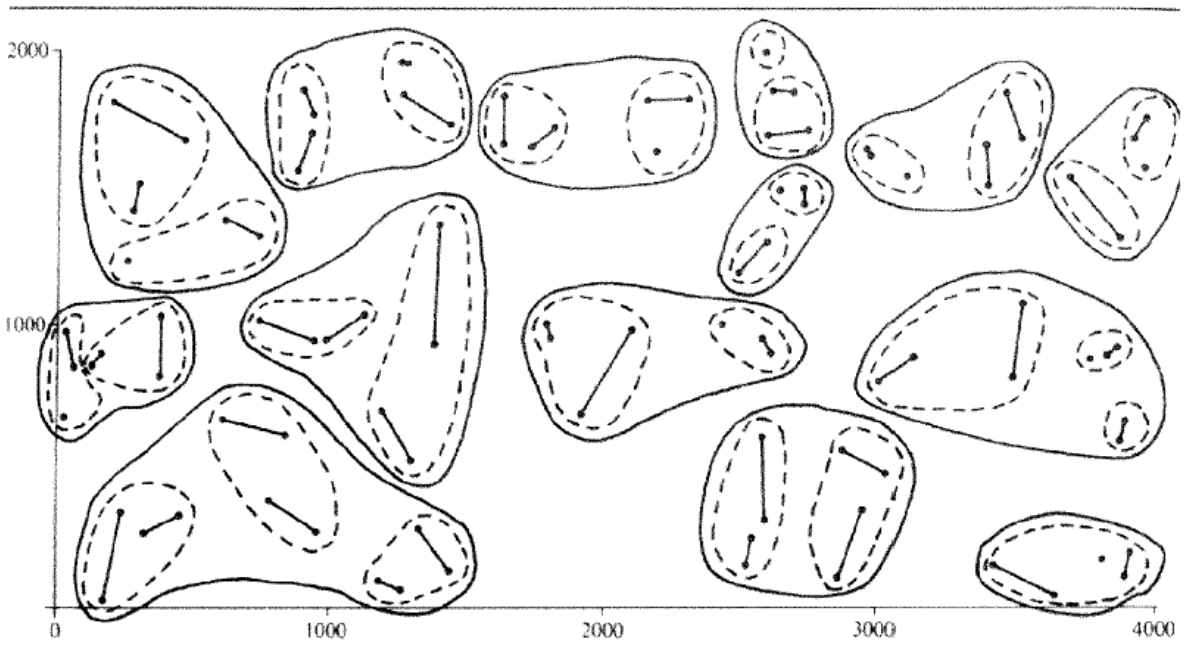
1968 Michie, Oldfield a Fleming [10]
Jedna z prvých štúdií na túto tému, ktorej výsledkom bolo, že človek vo väčšine prípadov podal porovnateľný a v jednom prípade dokonca lepší výkon, ako vtedajšie algoritmy pre vyhľadávanie v grafoch.

1971 Krolak, Felts, Marble [7]
V tejto štúdii sa autori zaoberajú možnosťou kombinácie počítačového a ľudského potenciálu prie riešení euklidovského TSP. Podarilo sa im navrhnúť postup, ktorý vedie k efektívnemu riešeniu problému pre inštancie o veľkosti 100 až 200 vrcholov. Dôvod, ktorý ich k tomu viedol, je, že pri veľkých počtoch vrcholov ani vizuálne podané zadanie nedá človeku dostatok informácií pre vyriešenie problému.

Základná myšlienka spočíva v tom, že počítač svojou výpočtovou silou pripraví pre človeka vizuálne prezentovanú informáciu o grafe, podľa ktorej sa človek rozhodne, ktoré vrcholy budú s ďalšom kroku spojené a ktoré nie. Počítač vždy pripravuje viacero návrhov a človek by z nich mal vybrať ten najlepší.

Postup použitý pri riešení problému bol nasledovný:

- Viacnásobným použitím algoritmu pre problém priradenia počítač vypočíta oblasti vrcholov, ktoré ležia blízko seba a vo výslednej kružnici by mohli byť prepojené navzájom. Tento výpočet sa aplikuje viackrát vždy na výsledok predošlého výpočtu. Výsledok bude podobný ako na obrázku 1.
- Človek vygeneruje svoje vlastné riešenie problému (je možné použiť na riešenie aj viacero ľudí paralelne).
- Použitím známych algoritmov pre riešenie TSP počítač vypočíta viacero riešení.
- Všetky tieto informácie sa dajú k dispozícii vo vizuálnej podobe človeku.
- Na základe svojej schopnosti nadhľadu človek vytipuje oblasti, kde by sa dal výsledok zlepšiť (tieto oblasti mu môže ponúknuť aj počítač s pomocou už známych procedúr).
- Následne sa spustia procedúry pre lokálne zlepšenie výsledku pre dané oblasti.
- Človek podľa vizuálnej informácie skombinuje ponúknuté riešenia do výsledku.
- Výsledok je možné zase rovnakým postupom zlepšiť, odhad, či sa ešte výsledok oplatí zlepšovať alebo nie, je úlohou človeka.



Obrázok 1 Príklad návrhu oblastí v grafe pri spoločnej práci človeka a algoritmu

Vzhľadom na časové obdobie, kedy experiment prebiehal (rok 1971), autori ohodnotili hodinu práce človeka ako ekvivalent jednej minúte práce počítača.

Výsledky experimentu ukázali, že pri spomínanom modeli spolupráce boli problémy o veľkosti 100 až 200 vrcholov vyriešené za menej ako 90 minút. Autori porovnávali výsledky s už známymi výsledkami pre použité grafy a okrem jedného prípadu bolo vždy nájdené optimálne riešenie. Pri náhodne vygenerovaných problémoch použili vtedajšie odhady pre vlastnosti optimálnych riešení.

Tento experiment v sebe okrem užitočných testovacích dát obsahuje zaujímavú myšlienku a to je dodať človeku nejakú počiatočnú informáciu o grafe, pre ktorý ma zostrojiť minimálnu hamilntonovskú kružnicu. Napríklad ako pomoc môže slúžiť množina riešení pre daný graf, ktoré budú výsledkom nejakého časovo nie náročného algoritmu. Problém je v tom, či tieto informácie zásadne neovplyvnia postup človeka pri riešení problému a či tento nebude chcieť „napodobňovať“ použitý algoritmus. Tento postup taktiež otestujeme v experimentálnej časti.

1996 MacGregor, Ormerod [9]

Motiváciou tejto štúdie, ako autori popisujú na začiatku, nebolo zlepšiť výkon algoritmov pre riešenie problému, ale zistiť a popísať schopnosti človeka riešiť tento problém. Autori popísali množinu experimentov, za účelom otestovania hypotézy, podľa ktorej efektivita ľudského riešenia TSP závisí od toho, v akom pomere je počet vrcholov konvexného obalu množiny vrcholov ku celkovému počtu vrcholov. Táto myšlienka je motivovaná výsledkom, ktorý publikoval Flood v roku 1956 [4], kde ukázal, že optimálne riešenie euklidovského TSP má nasledujúcu vlastnosť:

Body, ktoré patria do konvexného obalu grafu sú v optimálnom riešení zoradené v poradí, v akom tvoria konvexný obal. Riešenie, ktoré porušuje toto pravidlo, teda nie je optimálnym riešením.

Vo svojom experimente autori prišli k záveru, že človek je schopný tým ľahšie nájsť efektívne riešenie, čím viac vrcholov v inštancii problému patrí do konvexného obalu.

Na hodnotenie kvality ľudského riešenia boli použité tri metódy:

- Kvalita riešenia sa hodnotila na základe jeho odchýlky od optimálneho riešenia. Toto hodnotenie však nie je možné použiť vo všeobecnosti, ale len pre danú inštanciu problému pre porovnanie použiteľnosti viacero metód pre túto inštanciu. Ak nie je k dispozícii optimálne riešenie, vyberie sa najlepšie známe riešenie.

- Druhá metóda je založená na štatistickom hodnotení a priemernej odchýlke od strednej hodnoty rozdelenia všetkých možných dĺžok ciest pre daný graf.

- Tretí prístup porovnáva ľudské riešenia s výsledkami troch heuristických procedúr. Bola použitá metóda Hľadanie najbližších susedov (Nearest Neighbour Search), a dve metódy vkladania vrcholov do konvexného obalu. Vkladanie podľa najväčšieho vnútorného uhla (Largest interior angle) a vkladanie s použitím kritéria najlacnejšieho vkladu (Cheapest insertion criterion).

Experiment bol rozdelený na dve časti.

V prvej časti boli použité rovinné grafy s desiatimi vrcholmi. Bolo použitých 7 zadaní. Počet vnútorných vrcholov prvých 6 zadaní bol od 1 do 6, pričom vonkajšie vrcholy boli usporiadané do pravidelného mnohouholníka. Vnútorné vrcholy boli generované náhodne, s určitými obmedzeniami, napríklad museli mať určitú minimálnu vzdialenosť od stredu mnohouholníka. Ako siedmy graf bol použitý graf z odbornej literatúry [1] (ďalej Dantzig graf), ktorý bol definovaný pre testovanie heuristických metód, keďže sa ukázalo, že väčšina aj pomerne kvalitných algoritmov na ňom zlyháva pri hľadaní optimálneho riešenia. Ako testovacie subjekty boli vybraných 58 študentov univerzity, na ktorej experiment prebiehal. Študentom boli problémy predložené na papieri, ktorý obsahoval vytlačené body. Čas na riešenie jedného zadania bol určený na päť minút.

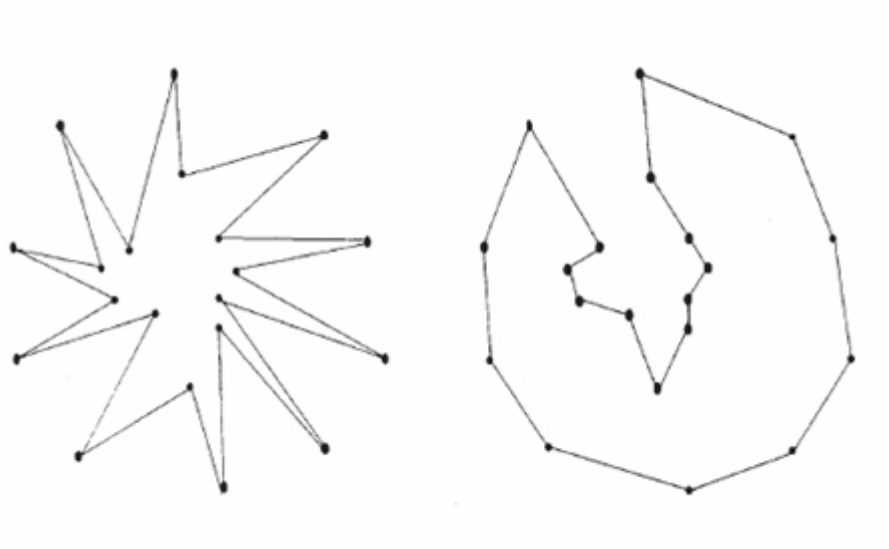
Výsledky prvej časti experimentu boli štatisticky vyhodnotené. Vykazovali dve dôležité vlastnosti:

- Študenti boli pri riešení problému úspešnejší ako tri použité algoritmy pre hľadanie optimálneho riešenia. Pre Dantzig graf sa trom študentom podarilo nájsť optimálne

riešenie, čo sa to nepodarilo ani jednému z troch použitých algoritmov. V priemere boli riešenia študentov od 1% do 3% bližšie optimálnemu riešeniu, ako riešenia použitých algoritmov.

- Po analýze riešení študentov sa zistilo, že miera rozdielov v riešeniach jednotlivých študentov a tým aj miera optimálnosti riešenia bola v korelácii s počtom vnútorných vrcholov v zadaniach. Čím väčší bol počet vnútorných vrcholov, tým viac sa pre daný graf riešenia jednotlivých študentov navzájom odlišovali, a tým viac sa riešenia študentov odlišovali od optimálneho riešenia.

- Ďalšie zaujímavé pozorovanie je, že vo výsledkoch, ktoré boli bližšie optimálnemu riešeniu, bol menší počet spojení vrcholov konvexného obalu s vnútornými vrcholmi.



Obrázok 2 Príklad na rôzny počet spojenia vnútorných bodov s bodmi konvexného obalu pre ten istý graf

Druhá časť experimentu prebiehala podobne, len s tým rozdielom, že študentom boli predložené zadania obsahujúce 20 vrcholov. Testovaných bolo 29študentov. Počty vnútorných vrcholov sa pohybovali od 4 po 16. Zadaní bolo rovnako sedem ako v predošlom prípade. Jediné naviac použité obmedzenie bolo, že vnútorné vrcholy, keďže ich bolo viac, nemohli byť príliš blízko seba. Zadania boli rozdané študentom rovnakým spôsobom, ako v predošlom prípade.

Výsledky boli podobné ako v prvom prípade. Rozdiely medzi odchýlkami od optimálneho riešenia pri študentoch na jednej strane a použitých algoritmoch na druhej strane boli

vyššie a pohybovali sa v rozmedzí 2-9%. Tak isto podporovali hypotézu, že kvalita ľudského riešenia problému výrazne závisí od počtu vnútorných bodov v zadanom grafe.

Závery práce sú pomerne obšírne, na podporu skúmanej hypotézy uvádzajú aj nepublikovaný článok, v ktorom boli analyzované problémy o rozmeroch od 10 do 60 vrcholov a vykazovali podobné výsledky ako popísaný experiment. Rozdiely medzi ľudským a optimálnym riešením záviseli do väčšej miery od počtu vnútorných vrcholov, než od celkového počtu vrcholov.

V záverečnej diskusii je okrem iného spomenuté, že v prípade zadania formou incidenčenej tabuľky so zadanými vzdialenosťami bodov by sme sa v prípade ľudského výkonu s veľkou pravdepodobnosťou nedočkali tak kvalitného výsledku ako pri grafickej reprezentácii zadania. Tvrdenie je podporené publikovaným experimentom [11], v ktorom sú výsledky pre tabuľkové zadanie grafu kvalitatívne oveľa horšie. To znamená, že vizuálna stránka veci je pri riešení problému človekom zrejme rozhodujúca.

Dôležitou informáciou uvedenou v záveroch tejto práce je, že študenti pri riešení problému vo veľkej väčšine prípadov pospájali do svojho riešenia body konvexného obalu v poradí za sebou., to znamená, že vo výslednej ceste sa nachádzali v poradí tak, ako stáli v pravidelnom mnoho uholníku buď v smere alebo proti smeru hodinových ručičiek. Čiže v podstate bez hlbších vedomostí o probléme intuitívne postupovali podľa zásady určenej Floodom v roku 1956 [4]. Tak isto je zaujímavé, že filozofia, ktorou postupovali pri riešení problému bola najbližšie jednoduchému greedy algoritmu, ktorý tvorí kružnicu tak, že vždy zaradí najbližšieho suseda.

Výsledky tohto experimentu poskytujú dôležité podklady pre túto prácu. Podporujú tvrdenie, že človek vo všeobecnosti lepšie rieši vizuálne zadaný TSP problém, ako testované algoritmy.

2006 Dry, Lee, Vickers, Hughes [3]

Tento experiment mal za úlohu zistiť časovú zložitosť ľudského riešenia problému. Autori nadviazali na podobný experiment z roku 2000 [5]. Veľký dôraz v tomto experimente sa kladie na štatistickú dôveryhodnosť experimentu.

Autori použili ako vstupy pre riešenie grafy s 10, 20,.. 120 vrcholmi vybrané z 10000 náhodne vygenerovaných vzoriek. Dôležité bolo, aby obsahovali pokiaľ možno reprezentačnú vzorku problémov aj vzhľadom na vlastnosti, ktoré ovplyvňujú zložitosť riešenia pre človeka, t.j. počet bodov konvexného obalu a počet možných priesečníkov spojníc dvoch vrcholov v grafe (vychádza z predchádzajúcej štúdie z roku 2000´[4]). Dosiahnuté výsledky sú porovnávané s algoritmami na riešenie problému. Ide o algoritmus vkladania do konvexného obalu [9] z predošlého experimentu, hierarchický prístup publikovaný Grahamom a kolektív v roku 2000 [4] a piatimi algoritmami prebratými z oblastí umelej inteligencie a operačného výskumu.

Účastníci experimentu riešili úlohu priamo na počítači, pričom dostávali spätnú väzbu o kvalite ich riešenia vo forme percentuálnej odchýlky od odhadovaného optimálneho riešenia vypočítaného metódou simulovaného žíhania.

Výsledkom štatistického vyhodnotenia bolo, že riešitelia problému ho dokázali riešiť v lineárnom čase. Čo sa týka časovej zložitosti riešenia prekonali ostatné testované algoritmy. Na druhej strane, pri vyšších časových nákladoch niektoré testované algoritmy dosahovali lepšie výsledky.
Keďže človek pri riešení problému dosahuje lineárnu časovú zložitosť, je zrejmé že pri riešení problému postupuje lineárne. V štúdii bol tento prístup nazvaný algoritmus hierarchického hľadania najbližšieho suseda (hierarchical nearest neighbour-HNN)[3]. Hľadaním najbližšieho suseda sa vytvoria skupiny navzájom prepojených vrcholov a v ďalšom kroku sa hľadá prepojenie týchto skupín.

V závere autori odporúčajú tvorcom moderných algoritmov zamerať sa na spôsob ľudského riešenia problému TSP.

4. Postup práce

Všetky preštudované články a experimenty v tejto oblasti boli motivované dvoma základnými cieľmi. V prvom prípade bol zámerom psychologický výskum za účelom analýzy ľudského myslenia a vnímania pri riešení vizuálne podaného problému[5], [8], [9], [11]. V druhom prípade išlo o porovnanie výkonov počítača s výkonmi človeka a stanovenie časovej zložitosti pre riešenie problému človekom [1], [3], [8], [10]. Jedna štúdia sa zaoberala kombináciou práce človeka a počítača [7].

V tejto práci je zámer vedomosti o riešení problému človekom použiť pre zostrojenie algoritmu, ktorý bude riešiť problém samostatne bez ďalšieho prispenia človeka. Jeho výkon sa následne pokúsime overiť na čo najväčšom množstve experimentálnych dát. Algoritmus by nemal byť založený na učení, alebo lokálnom vylepšovaní rýchlo vypočítaného suboptimálneho riešenia, ale na znalosti problému ako takého.

Vzhľadom na tento cieľ použijeme nasledovný postup:

- Vytvorenie aplikácie, ktorá umožňuje človeku riešiť vizuálne zadaný E-TSP problém na počítači.
- Distribuovanie aplikácie ľuďom z môjho okolia a analýza ich postupov a postrehov pri riešení problému.
- Zhodnotenie výsledkov a postupov použitých riešiteľmi pri riešení problému.
- Na základe výsledkov návrh jedného, prípadne viac algoritmov pre riešenie problému.
- Odhad a dôkaz ich zložitosti.
- Test algoritmov na čo najväčšom počte vstupných dát a vyhodnotenie výsledkov.

5. Praktické spracovanie

5.1. Pracovná aplikácia

Na vytvorenie aplikácie bol použitý programovací jazyk Delphi, vývojové prostredie Borland Deplhi 7.0. Aplikácia pracuje v prostredí MS Windows.

Používateľské rozhranie aplikácie sa skladá z troch okien. Prvé okno slúži na zadávanie vrcholov grafu, klikaním myši do plocha okna a následné spúšťanie algoritmov. Ďalej obsahuje funkcie pre odstraňovanie hrán a odstraňovanie celého grafu.
Pre daný graf je možné spustiť kompletné prehľadávanie a algoritmus hľadania najbližšieho suseda (ďalej len NS algoritmus). Pre algoritmus kompletného prehľadávania je použité generovanie permutácií bez optimalizácie, takže vo vyhovujúcom čase tento postup nachádza riešenia len pre grafy do 15 vrcholov.

NS algoritmus je jednoduché hľadanie najbližšieho suseda, pracujúce v kvadratickom čase. Do aplikácie bol zaradený jednak kvôli porovnaniu s ďalej navrhovanými algoritmami a tak isto aj za účelom získania odhadov na vlastnosti riešenia pre grafy s väčším počtom vrcholov, pre ktoré nebolo možné spustiť kompletné prehľadávanie. Pre tento algoritmus som zaradil aj možnosť výberu vrcholu, z ktorého má výpočet začínať.
Oba algoritmy po ukončení výpočtu vypíšu postupnosť vrcholov, znázornia hrany v grafickom okne a vypíšu celkovú dĺžku vypočítanej kružnice.

Pre porovnanie výsledkov so simuláciou ľudského postupu bolo navyše pridané aj tlačidlo pre lokálnu optimalizáciu, ktorá bola pôvodne určená ako súčasť algoritmu lineárneho prehľadávania.

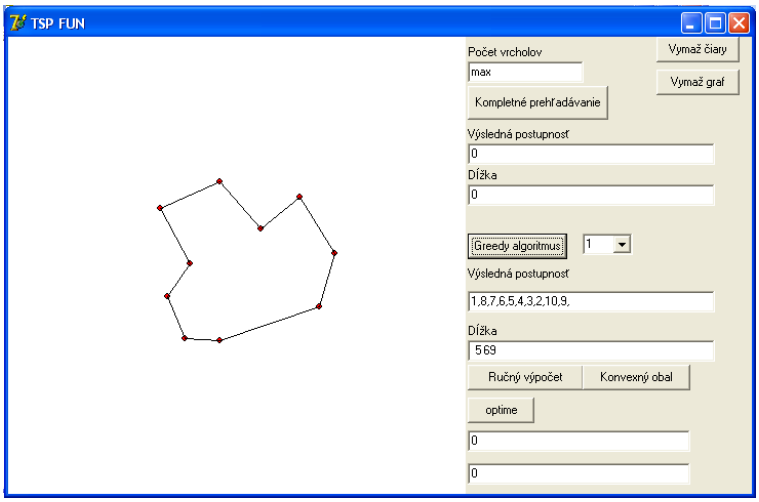
Grafy do aplikácie sú zadávané používateľom klikaním myši do na to určeného poľa. Takto zadávané grafy je možné považovať za náhodne generované, ak, samozrejme, zadávateľ úmyselne alebo podvedome nepridáva vrcholy vždy rovnakým spôsobom. Vrcholy grafov sú reprezentované červenými kruhmi s takým priemerom, aby bolo možné pohodlne na ne klikať myšou.

Okrem toho bolo do aplikácie „napevno“ vložených niekoľko grafov kvôli testovaniu výkonov a algoritmov na problémoch, ktoré sa považujú za bežnými algoritmami ťažko riešiteľné a používajú sa na hodnotení výkonnosti a kvality.

Na hlavnom okne sú ďalej tlačidlá spúšťajúce ďalšie dve okná („ručný výpočet“ a „konvexný obal“).
Prvé z nich slúži na manuálne riešenie problému zadaného v hlavnom okne. Ručný výpočet pri spustení skopíruje z prvého okna len zadanie grafu. Následne umožní používateľovi spájať vrcholy hranami (klikaním myši) a vypisuje celkovú dĺžku doteraz vytvorenej trasy. Takýmto spôsobom môže používateľ vytvoriť vlastné riešenie a porovnať ho s výsledkami NS a kompletného prehľadávania. Spojnice je možné zmazať a začať spájanie odznova. Do tohto okna bol neskôr pridaný aj algoritmus, ktorý bol navrhnutý podľa pozorovania ľudského postupu, o ktorom bude reč ďalej.

Druhé okno slúži na výpočet postupnosti konvexných obalov pre zadaný graf a zároveň ako vstupná informácia pre navrhovanú metódu spájania konvexných obalov, ktorá bude popísaná ďalej.

Pri vývoji aplikácie bol kladený dôraz hlavne na praktické použitie aplikácie, keďže pre účel práce neboli dizajn a robustnosť až také podstatné. To znamená, že aplikácia predpokladá, že používateľ sa bude správať „logicky“ a postupovať po poradí tak ako majú operácie význam. Keďže aplikácie nie je určená na distribúciu ani prezentáciu, z praktického hľadiska ju považujem za dostačujúcu. Rozhranie aplikácie je na nasledujúcom obrázku:



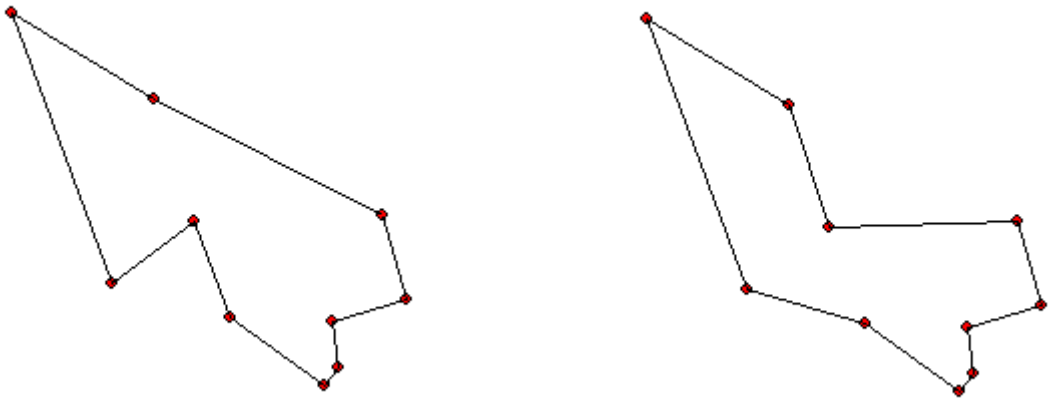
Obrázok 3 Hlavná pracovná plocha aplikácie

5.2. Experiment

Aplikáciu bola odovzdaná vybraným účastníkom z môjho okolia. Keďže nešlo o psychologický alebo štatistický výskum, nebol náhodný výber participantov až tak dôležitý. Potrebné skôr bolo, aby participanti mali určité analytické a logické myslenie, potrebné pre riešenie takéhoto problému.
Účastníci boli poväčšine ľudia z IT prostredia.

Z prieskumu postupov pri riešení problému vyplynuli nasledujúce pozorovania:

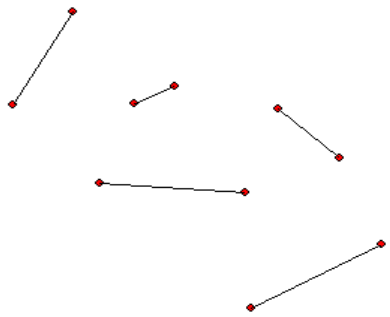
- Najčastejší postup pri riešení problému spočíval v lineárnej technike pridávania najbližšieho vrcholu k naposledy pridanému, s ohľadom na najbližšie okolie pridaného vrcholu. To znamená, že skôr, ako človek vrchol pridal, preskúmal vizuálne jeho najbližšie okolie. V mysli prehodnotil možnosti a podľa toho pridal nasledujúci vrchol. V podstate by sa tento postup dal nazvať metódou najbližšieho suseda s lokálnym prehľadávaním. Úspešnosť postupu závisela hlavne na tom, či bol graf zadaný dostatočne prehľadne, to znamená, či boli dostatočne viditeľné rozdiely vo vzdialenostiach bodov. Následne po prezretí hotovej kružnice bola urobená lokálna optimalizácia.



Obrázok 4 Príklad lokálnej optimalizácie pri postupe človeka

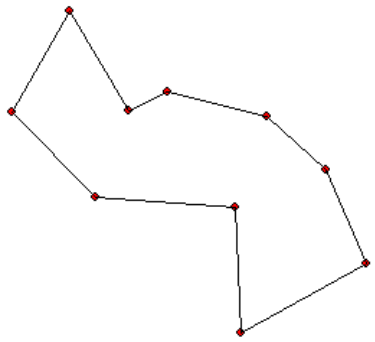
- Podľa očakávaní bol pre riešiteľa problém zložitejší, ani nie je tak pri väčšom počte vrcholov, ako pri vyššom počte vnútorných bodov (body, ktoré nepatria do konvexného obalu). Tento prístup bol najčastejší a odráža pravdepodobne najrozšírenejšie ľudské vnímanie tohto problému, čiže lineárny postup s lokálnym prehľadávaním a následnou revíziou výsledku. Podľa vyššie spomínaných prác vedie tento postup, z hľadiska časovej náročnosti, k pomerne dobrým výsledkom..

- Druhý postup použitý participantmi bol pomerne zriedkavý (dva prípady). Postup sa líši od predošlého len v prvej časti, kde boli pospájané vrcholy tak, že každý bol spojený najviac s jedným vrcholom. Ako vrchol pre vytvorenie dvojice bol vždy braný najbližší voľný vrchol. Ako prvé boli vybrané vrcholy, ktorých vzdialenosť bola najkratšia v grafe. Následne druhá najkratšia a tak ďalej. Po výbere dvojíc vyzeral graf ako na nasledujúcom obrázku:



Obrázok 5 Spájanie dvojíc najbližších vrcholov

Samozrejme je potrebné dávať pozor aby spojnice bodov neviedli krížom cez graf. Ak ostala posledná nespojená dvojica vrcholov každý na opačnej strane grafu, tak boli v prvej fáze nechané tak. Následne bol graf doplnený na kompletný okruh ako na nasledujúcom obrázku:



Obrázok 6 Dokončenie kružnice pri metóde spájania najbližších vrcholov

Táto metóda by sa dala nazvať vyhľadaním najkratších vzdialeností a následným spájaním komponentov, ktoré sú k sebe najbližšie, prípadne, ktoré sú v poradí.
V prípade tohto grafu to viedlo k úspechu, ale v globálnom merítku táto metóda nebola tak úspešná ako prvá popisovaná. Po dokreslení kružnice bolo nutné viac krát sa vrátiť a meniť počiatočné spojenie hrán. Toto umožňuje nadhľad človeka nad problémom, ale pri práci algoritmu by bolo nutné komplikované prehľadávanie. Z tohto pohľadu je prvý postup menej časovo náročný.

V použitej literatúre sa často spomína heuristika najlacnejšieho vkladania alebo vkladania podľa najväčšieho uhla. Ako počiatočný stav pre tento algoritmus v oboch prípadoch slúži konvexný obal množiny vrcholov. Autor programu sám prispel návrhom metódy riešenia problému TSP, nakoľko, z pochopiteľných dôvodov, bol on sám najčastejším testovacím subjektom tejto aplikácie.
Z autorových vlastných záverov vyplýva návrh algoritmu, ktorý bude pracovať podobne ako spomínané dva, lenže bude vždy vkladať do vnútra konvexného obalu len body zo susedného konvexného obalu o jednu úroveň vyššie. Algoritmus je bližšie popísaný v ďalšej časti práce.

5.3. Návrh algoritmov pre riešenie problému

5.4. Algoritmus lineárneho prehľadávania

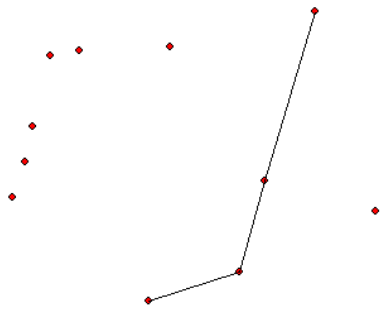
Vlastnosť, ktorou sa ľudské prehľadávanie grafu líši od algoritmu hľadania najbližšieho suseda, je, že človek začne prehľadávať graf jedným smerom a ten dodržuje. Prehľadáva graf v podstate „dokola“, a preto sa mu nestane, že by porušil pravidlo, že body konvexného obalu musia byť vo výslednej kružnici zoradené v poradí, v akom tvoria konvexný obal. Tak isto sa mu nestane, že by vytvoril dve križujúce sa hrany. To znamená, že v navrhovanom algoritme je potrebné postupovať po smere alebo proti smeru hodinových ručičiek, smer nemeniť a testovať, či sa práve pridávaná hrana nekrižuje s nejakou už pridanou. V tomto ohľade je postup podobný, ako pri hľadaní konvexného obalu. Takže prvé vylepšenie prehľadávania bude spočívať v dodržovaní smeru prehľadávania. To zabezpečíme zavedením orientácie. Budeme si počas výpočtu udržiavať informáciu o tom, ktorým smerom po kružnici sa pohybujeme. Pri každom pridávanom vrchole budeme následne testovať, či sa práve pridávane hrany nepretínajú s nejakou predtým vytvorenou. Keďže sa nepodarilo interpretovať verne ľudské prehľadávanie grafu, zvolili sme postup s vkladaním vrcholov a následnou lokálnou optimalizáciou. Táto by mala zastupovať následnú vizuálnu kontrolu výsledného grafu človekom.

5.5. Postup

V algoritme budeme postupovať vždy proti smeru hodinových ručičiek. Začiatok súradnicovej sústavy je v ľavom hornom rohu obrazovky a celý postup je popisovaný z pohľadu používateľa.

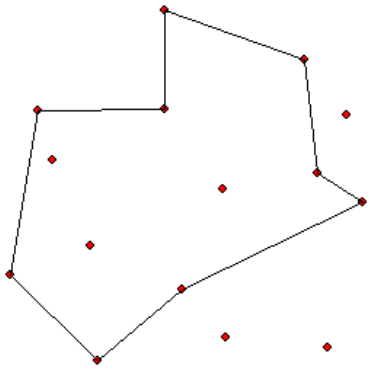
1. V prvej fáze nájdeme vrcholy s maximálnymi a minimálnymi súradnicami, teda najvrchnejší, najspodnejší, najviac vpravo a najviac vľavo. V prípade, že pre každý typ je takýchto vrcholov viac, zoradíme ich do postupnosti podľa druhej súradnice tak, aby to zodpovedalo smeru prehľadávania.
2. Začneme od vrcholu s najväčšou y súradnicou a postupujeme proti smeru hodinových ručičiek. Ak je takých vrcholov viacero, tak začneme od vrcholu s najmenšou x súradnicou..

3. Hľadáme vždy najbližší vrchol umiestnený smerom vpravo a nahor od aktuálneho vrcholu. Ak je takých vrcholov viac, vezmeme ten, ktorý je bližšie ku okraju grafu. Pri postupe doprava nahor to bude vrchol s najmenšou *y* súradnicou.
4. Takto postupujeme, až kým neprídeme na bod s maximálnom *x* súradnicou. Pri hľadaní robíme nasledovné testy:
    a. Či vrchol, ktorý sme práve pridali, nebude meniť smer prehľadávania, teda či nie je okrajový (v tomto prípade bod s maximálnou *x* súradnicou).
    b. Či vrchol, ktorý pridávame, nemá menšiu *y* súradnicu ako vrchol (alebo vrcholy, ak je ich viac) najviac vpravo. Situácia je znázornená na nasledujúcom obrázku. Týmto zabezpečíme, že vrchol, ktorý je najviac v pravo bude určite zaradený do postupnosti.



Obrázok 7 Príklad chybného postupu pri lineárnom prehľadávaní

    c. Či vrchol, ktorý pridávame nevytvorí s naposledy pridaným hranu, ktorá pretína niektorú z doteraz vytvorených hrán.
5. Pri práci udržiavame vždy dve disjunktné skupiny vrcholov, z jednej vyberáme, do druhej pridávame.
6. Takto sa dostaneme k vrcholu s maximálnou *x* súradnicou.
7. Ak je vrcholov s maximálnou *x* súradnicou viac, pridáme ten z nich, ktorý má najväčšiu *y* súradnicu a nastavíme zmenu smeru.
8. Ďalej postupujeme analogickým spôsobom v smeroch doľava a hore, doľava a dole respektíve doprava a dole.
9. Kružnicu uzatvárame v stave, keď už nie je možne pridať vrchol pri ceste doprava dole. Posledný vrchol spojíme s prvým vrcholom postupnosti.
10. Výsledkom bude kružnica blízka konvexnému obalu, ktorá však vo všeobecnosti nebude obsahovať všetky vrcholy grafu (pozri nasledujúci obrázok)



Obrázok 8 Výsledok prvej fázy lineárneho prehľadávania

11. Následne budeme prechádzať po hranách vytvorenej trasy a budeme vkladať ďalšie vrcholy. Pri vkladaní vrcholu zrušíme hranu kružnice tvorenú dvoma vrcholmi, medzi ktoré vrchol vkladáme a pridáme dve hrany spojením vkladaného vrcholu s vrcholmi odstránenej hrany. Vkladaný vrchol musí mať nasledovné vlastnosti:
    a. Pre danú hranu musí jeho pridanie spôsobiť najmenší možný nárast dĺžky kružnice zo všetkých voľných vrcholov.
    b. Vloženie vrcholu nebude spôsobovať menší nárast dĺžky kružnice pre žiadnu inú hranu kružnice.
    c. Pridaním vrcholu do kružnice nesmú vznikať kríženia hrán.

    V danej situácii takto vložený vrchol predstavuje najmenší možný nárast dĺžky kružnice pri jeho vložení a pri dodržaní podmienky nekríženia sa hrán. Tento postup môže vyžadovať viacero prechodov kružnice, kým budú zaradené všetky vrcholy. (ak sa pri vkladaní dostaneme na koniec kružnice, začneme ju prechádzať odznova) Pokiaľ bude vrchol vykazovať rovnaký nárast dĺžky kružnice pri vložení na dve rôzne miesta, vložíme ho medzi tie dva vrcholy, ktoré sú bližšie k začiatku kružnice. Keď zaradíme posledný vrchol, prejdeme k ďalšej fáze.

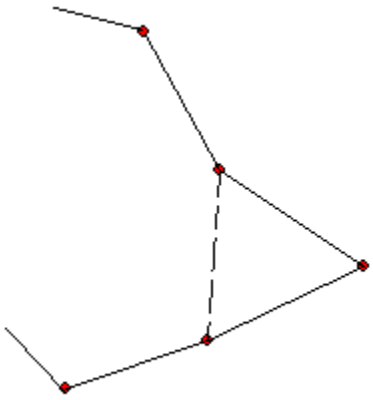
12. Po zaradení všetkých vrcholov urobíme jednoduchú optimalizáciu:
    a. Prechádzame všetky vrcholy v poradí, v akom sú zoradené v kružnici.
    b. Každý vrchol z kružnice vyberieme.
    c. Následne ho vložíme do kružnice naspäť, tak aby bol nárast dĺžky kružnice minimálny.
    d. Vždy, keď sa poradie vrcholov zmení, začneme celý proces odznova od prvého vrcholu.

Tento postup budeme opakovať dovtedy, kým sa bude dĺžka kružnice zmenšovať. Keď sa po prechode celej kružnice dĺžka kružnice nezmení, skončíme. Popisovaná metóda zabezpečuje, že postupnosť vrcholov zmeníme len vtedy, ak sa tým skráti kružnica. Tento krok má nahradiť lokálnu optimalizáciu pri práci človeka.

13. V poslednom kroku algoritmus vykreslí výsledok na obrazovku.

Teraz je potrebné ukázať, že výsledkom algoritmu bude hamilntonovská kružnica pre daný graf spĺňajúca podmienku o usporiadaní vrcholov konvexného obalu..

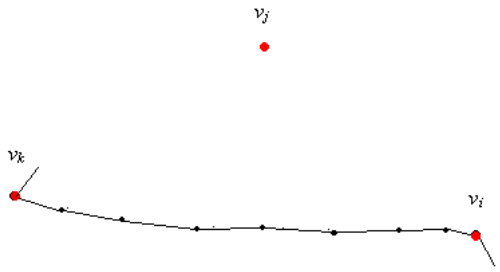
- Pridanie prvého bodu je v poriadku. Následne pri postupe smerom doprava a nahor tvoríme jednoduchú, súvislú cestu, v ktorej žiadny vrchol nie je dvakrát, a je konečná. Jej maximálna dĺžka je ohraničená počtom vrcholov grafu. Podobne to platí aj o nasledujúcich krokoch. (postupne cez všetky body s maximálnymi súradnicami) V poslednom kroku sa pripojí bod, z ktorého cesta začínala. Takže výsledkom bude kružnica, ktorá ale nemusí obsahovať všetky vrcholy. Keďže nové vrcholy pridávame vždy tak, aby sa vzniknuté hrany nekrížili, výsledná kružnica nebude obsahovať križujúce sa hrany.
- Pri pridávaní zvyšných vrcholov udržujeme kružnicu. Vyhľadáme hranu a vrchol na pridanie, zrušíme hranu a jej dva vrcholy spojíme s pridávaným vrcholom (pozri nasledujúci obrázok). Tým kružnica ostane zachovaná a zároveň zabránime kríženiu hrán. Voľný vrchol vieme vždy pridať do kružnice. Tento postup tak isto skončí, keďže voľných vrcholov je konečný počet.



Obrázok 9 Vkladanie vrcholu do kružnice

- Nasleduje optimalizácia. Každým vybraním vrcholu a jeho pridaním naspäť ostane zachovaná kružnica, pričom tento proces je konečný, nakoľko vďaka podmienke vkladania vrcholu metódou najlacnejšieho vloženia je celková dĺžka kružnice nerastúca

Formulujeme nasledujúce tvrdenie:
Ak kružnica nerešpektuje pravidlo o usporiadaní vrcholov konvexného obalu, potom musí obsahovať krížiace sa hrany.
Dôkaz:
Predpokladajme, že $v_1, v_2,...,v_m$ je postupnosť vrcholov konvexného obalu, kde $m \geq 3$ (v trojuholníku existuje len jedno možné poradie). Majme tri susedné vrcholy v_i, v_j a v_k, ktoré sa v obale nachádzajú v tomto poradí. Nech sa vo výslednej kružnici nachádzajú v inom poradí, napríklad v_i, v_k, v_j (vždy sa jedná o usporiadanie v kladnom smere). Vezmime si časť kružnice, ktorá vedie z v_i do v_k. Tento oblúk oddeľuje vrchol v_j od ostatnej časti grafu. Ak ho chceme spojiť so zvyškom grafu do kružnice, tak táto spájajúca cesta musí niektorou svojou hranou pretínať oblúk v_i, v_k. A teda kružnica obsahuje križujúce sa hrany (pre lepšiu ilustráciu pozri obrázok nižšie



Obrázok 10 Oddelenie vrcholu konvexného obalu od zvyšku grafu spôsobujúce kríženie hrán

Z predošlého tvrdenia následne vyplýva, že ak kružnica neobsahuje vzájomne pretínajúce sa hrany, tak vrcholy konvexného obalu budú v kružnici zoradené v poradí tak, ako tvoria konvexný obal. Výsledok nášho algoritmu túto podmienku teda spĺňa.

5.6. Zložitosť algoritmu

- Inicializácia premenných a prípravné výpočty, napríklad výpočet vzdialeností pre incidenčnú maticu, trvá vzhľadom na použité dátové štruktúry $O(n^2)$.
- Extrémne body vieme nájsť v čase $O(n)$ jednouchým prehľadaním poľa vrcholov.
- Utriediť extrémne body do postupností trvá maximálne $O(n \log n)$ pri použití optimálneho triedenia.
- Pridanie jedného vrcholu do vytváranej kružnice trvá aj s testami maximálne $O(n^3)$ Vzhľadom na špecifickosť každého testovaného grafu nie je možné vo všeobecnosti určiť „priemernú“ dĺžku vytváranej kružnice, no vždy ju vieme ohraničiť počtom vrcholov daného grafu. Tu predpokladáme, že takto vytvorená kružnica bude mať k vrcholov. Potom súčet potrebného času pre najhorší prípad je

  $2\sum_{i=1}^{k} (n - i)^2$, kde k môže byť 3, 4, ..., n.

  - Prejdenie poľom nepridaných vrcholov a nájdenie vhodného vrcholu podľa smeru a vzdialenosti bude trvať čas $n - i$, kde i je počet doteraz zaradených vrcholov.
  - Podľa zvoleného smeru prechádzania grafu otestujeme, či kandidát neprekračuje extrémny vrchol v čase $O(1)$.
  - Test, či kandidát nevytvára hranu, ktorá by vytvárala kríženie s už pridanými hranami vykonáme v čase $n - i$.
  - Celý postup opakujeme k krát, kým nie je zaradených všetkých k vrcholov počiatočnej kružnice.
- Popridávanie vrcholov, ktoré ostávajú mimo počiatočnej kružnice, trvá najviac $O(n^4)$. Predpokladajme, že v prvom kroku vznikla kružnica dĺžky k a teda voľných vrcholov v grafe ostalo $n - k$, potom popridávanie zvyšných vrcholov bude trvať

  $2\sum_{i=0}^{n-k-1} (k + i)^2 (n - k - i)$, čo je v najhoršom prípade $O(n^4)$, ak bude k malé číslo v porovnaní s n.

  - Výpočet vzdialeností voľného vrcholu od k hrán a testy, či nevytvára kríženie hrán trvajú $2k^2$. Toto musíme urobiť pre každý voľný vrchol, aby

sme našli vrchol s minimálnym vkladom, čo spĺňa podmienku nepretínania sa hrán. Teda celkový čas pre pridanie jedného vrcholu je $2k^2(n-k)$.

- Celý postup sa vykoná maximálne $(n-k)$ krát (V najhoršom prípade, keď pri jednom prechode pridáme vždy len jeden vrchol).

- Optimalizácia bude trvať maximálne $O(n^3)$:
  - prechod k vrcholu trvá $O(n)$,
  - nájdenie hrany, od ktorej má vrchol najmenšiu vzdialenosť bez kríženia, bude trvať $O(n^2)$,
  - vloženie vrcholu $O(1)$,
  - celá procedúra sa potenciálne zopakuje pre každý vrchol.

Pre celkovú časovú zložitosť použitého algoritmu následne dostávame:

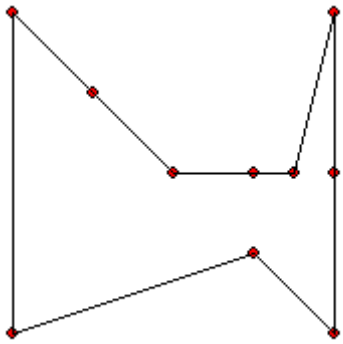
$$2\sum_{i=1}^{k}(n-i)^2+2\sum_{i=0}^{n-k-1}(k+i)^2(n-k-i)+O(n^3)$$

Ak k je malé číslo oproti n, tak z prvej sumy dostaneme $O(n^2)$ a z druhej $O(n^4)$. V opačnom prípade, ak sa nám v prvom kroku podarí vytvoriť čo najväčšia kružnica (dĺžkou blízka ku n), tak z prvej sumy dostaneme $O(n^3)$ a z druhej $O(n^4)$. V priemernom prípade, keď je k približne $\frac{n}{2}$, celková zložitosť bude tiež $O(n^4)$ kvôli hodnote druhej sumy.

Takže celková zložitosť algoritmu je v najlepšom prípade $O(n^3)$ a v priemernom a najhoršom prípade $O(n^4)$.

Priestorová zložitosť algoritmu bude $O(n^2)$, keďže používame konštantný počet lineárnych polí a jedno dvojrozmerné pre incidenčnú maticu s vzdialenosťami vrcholov. Použitiu tohto poľa by sme sa mohli vyhnúť za predpokladu, že by sme vždy, keď by to bolo potrebné, vypočítavali vzdialenosti medzi vrcholmi. Inicializácia a prípravné výpočty, napríklad výpočet vzdialeností trvá rovnako dlho.

Algoritmus bol testovaný na veľkom množstve náhodných grafov a na špeciálnom grafe, ktorý je používaný na testovanie algoritmov. Ide o Dantzig graf spomínaný v [1] zobrazený na obrázku nižšie aj s optimálnou kružnicou.



Obrázok 11 Optimálne riešenie pre Dantzig graf

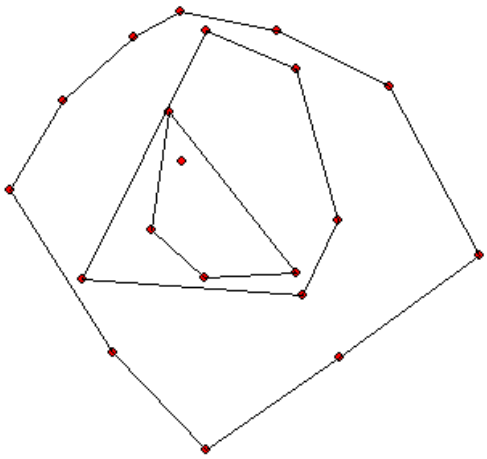
Pre potreby testovania bol algoritmus zaradený do aplikácie, pričom bola jeho činnosť rozdelená na vyššie popisované tri fázy, vytvorenie prvotnej kružnice, pridávanie zvyšných vrcholov a optimalizáciu, čo umožnilo lepšiu analýzu jeho činnosti.

5.7. Algoritmus postupnosti konvexných obalov

Tento algoritmus je navrhnutý nie základe experimentov, ale ako výsledok autorovho pozorovania a štúdia problému. Konvexný obal ako taký má veľký význam pri riešení tohto problému a viacero heuristík ho využíva ako počiatočný stav pri výpočte riešenia. Do kružnice vytvorenej konvexným obalom sa následne vkladajú vrcholy metódou najlacnejšieho vkladu alebo metódou najväčšieho uhla. V podstate sa jedná o greedy prístup.

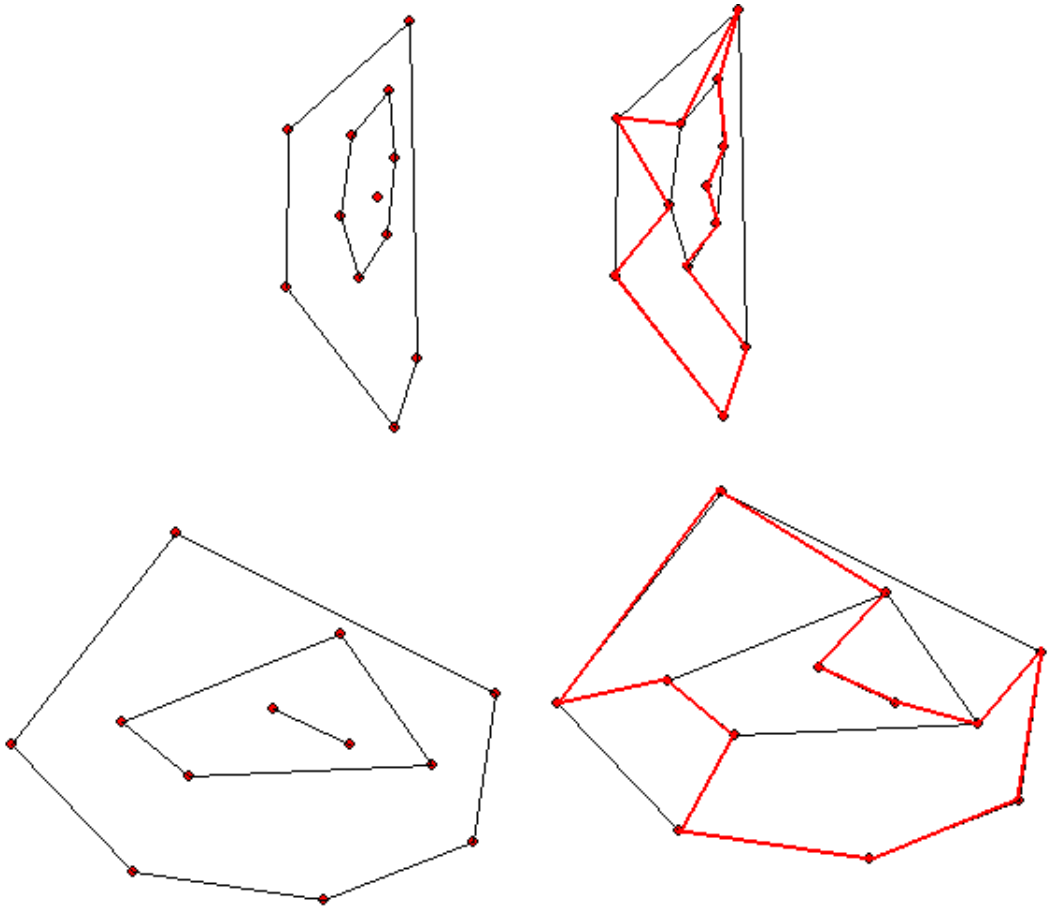
Pri testovaní rôznych grafov a ručného skúšania veľkého množstva postupov bola testovaná aj táto metóda pre získanie lepšieho prehľadu o probléme. Ako hlavný problém greedy algoritmov vo všeobecnosti sa javí ich prílišná „žravosť“. Bez ohľadu na budúci postup alebo bez ohľadu na vnútornú povahu problému sa akceptuje ako ďalší krok vždy najlepšie ponúkané riešenie. Aj bez dokonalej znalosti problému je intuitívne jasné, že zavedením určitej striedmosti by mohlo byť možné dospieť k lepšiemu riešeniu. Pri znalosti podstatných vlastností problému by dokonca mohlo byť možné nájsť optimálny prístup.

Na základe pozorovania a tejto úvahy rozdelíme graf na postupnosť konvexných obalov. Vytvoríme som konvexný obal grafu, tento z grafu vylúčime a zo zvyšku následne vytvorím ďalší konvexný obal. Takýmto postupom vznikla konečná postupnosť do seba vnorených a vzájomne disjunktných konvexných obalov (konkrétny príklad takejto postupnosti je zobrazený nižšie).



Obrázok 12 Príklad postupnosti konvexných obalov

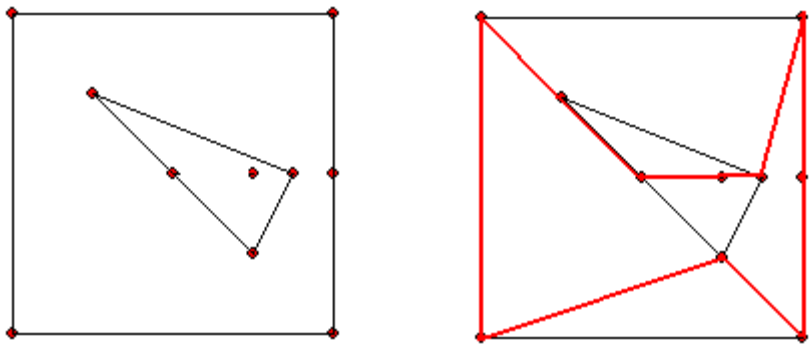
Za účelom nájdenia vzťahu medzi vytvorenou postupnosťou konvexných obalov a optimálneho riešenia TSP bolo vykonaných viacero pokusov. Pre lepšie priblíženie čitateľovi sme náhodne zvolili dva grafy (obrázok nižšie),na ktorých je demonštrovaný tento vzťah optimálnej kružnice a hierarchie konvexných obalov.



Obrázok 13 Príklady vzťahu optimálneho riešenia a hierarchie konvexných obalov

Na základe pozorovania je možné formulovať hypotézu, že pravidlo pre usporiadanie vrcholov konvexného obalu musí platiť aj pre podradené konvexné obaly. To by nám

dávalo ďalšie obmedzenie na vlastnosti riešenia. Toto tvrdenie však neplatí. Totiž ak aj kružnica obsahuje vrcholy niektorého z vnútorných obalov v inom poradí, k prekríženiu hrán nemusí dôjsť, kružnica môže prechádzať „obchádzkou“ cez nadradený konvexný obal, ako je možné vidieť pri optimálnom riešení pre Dantzig graf na nasledujúcom obrázku.



Obrázok 14 Optimálne riešenia a konvexné obaly pre Dantzig graf

Kružnica obsahuje vrcholy prostredného obalu v poprehadzovanom poradí. Dantzig graf potvrdzuje skutočnosť, že túto charakteristiku optimálnej kružnice nie je možné zahrnúť do návrhu tvorby hľadaného algoritmu.

Na ukážkových obrázkoch optimálna kružnica prechádza z jedného obalu len o jeden obal v hierarchii nižšie alebo vyššie. Ale nie je tomu tak. V niektorých prípadoch kružnica prechádza aj cez viacero obalov dovnútra, takže s touto vlastnosťou rovnako nie je možné pri navrhovaní algoritmu počítať.

5.8. Postup

Možností, ako spájať konvexné obaly, je viacero:

- Ako prvý vezmeme vonkajší obal a začneme k nemu pridávať postupne vnútorné obaly tak, že vždy pridáme obal o jednu úroveň nižšie.
- Ako prvý vezmeme najvnútornejší obal a postupujeme opačným smerom.
- Obaly spájame spôsobom vyhľadania vrcholu s najlacnejším vkladom a jeho pridania do vytváranej kružnice, alebo postupujeme po poradí v postupnosti

vrcholov vkladaného konvexného obalu a každý vkladaný vrchol vložíme najlacnejšie ako vieme

V nasledujúcom texte sa obmedzíme na jednu zo spomenutých metód, avšak nevieme vopred určiť, ktorá nám poskytne lepšie výsledky. Zvolíme metódu postupu smerom z vnútra von, s vkladaním vrcholu metódou najmenšieho nárastu dĺžky kružnice. Následne som nevolil žiadnu optimalizáciu v očakávaní, že algoritmus bude počítať pomerne dobré výsledky. Postup práce algoritmu je teda nasledovný:

1. Algoritmus vypočíta konvexný obal množiny vrcholov nejakou všeobecne známou metódou pre počítanie konvexného obalu. Pre tento prípad sme zvolili metódou Graham scan, ktorá pracuje v čase $O(n\log n)$, pričom sme ju prispôsobili aj pre špecifický prípad umiestnenia viacerých vrcholov na jednu priamku, a to bez zásadného vplyvu na jej časovú zložitosť.
2. Následne algoritmus vyradí vrcholy konvexného obalu z množiny vrcholov a vypočíta rovnakým spôsobom ďalší konvexný obal.
3. Konvexné obaly uloží do dvojrozmerného poľa pre rýchle spracovanie (tu bolo možné postupovať aj tak, že vždy vypočíta len jeden obal a ten pridá do priebežne vytváranej kružnice, ukladanie do poľa sme zvolili kvôli následnej vizualizácii postupnosti obalov).
4. Algoritmus zadefinuje vnútorný obal za doterajšiu kružnicu a postupne k nej pridáva vonkajšie obaly podľa poradia v hierarchii. Pri pridávaní vrcholu z obalu o úroveň vyššie k doteraz vytvorenej kružnice algoritmus postupuje nasledovne:
    - Nájde vrchol s najlacnejším vkladom, ktorý spĺňa podmienku nekríženia hrán.
    - Vrchol vloží do vytváranej kružnice podobne ako predchádzajúci algoritmus.
    - Postup opakuje pre všetky vrcholy obalu o úroveň vyššie.
5. Po pridaní posledného vrcholu vonkajšieho obalu algoritmus končí.
6. Následne program vykreslí vypočítanú kružnicu.

Teraz ukážeme, že algoritmus vypočíta hamilntonovskú kružnicu a odhadneme jeho časovú zložitosť.

- V prvých krokoch sa používa Graham scan, ktorý je všeobecne známy. Jeho správnosť tu z tohto dôvodu nebudeme dokazovať.
- Predpokladajme, že teda máme postupnosť konvexných obalov. Vybraný vnútorný obal tvorí hamilntonovskú kružnicu pre vrcholy tohto obalu.
- K nej pridávame vždy jeden vrchol z nadradeného obalu. Po pridaní ostane kružnica zachovaná a taktiež nevzniknú pretínajúce sa hrany.
- Vzhľadom na to, že obalov aj vrcholov je konečný počet, tento postup bude konečný.
- Teda výstupom algoritmu bude hamilntonovská kružnica pre daný graf, ktorá nebude obsahovať pretínajúce sa hrany a teda bude spĺňať aj podmienku o usporiadaní vrcholov konvexného obalu

5.9. Zložitosť algoritmu

- Inicializácia premenných a prípravné výpočty, napríklad výpočet vzdialeností pre incidenčnú maticu, trvá vzhľadom na použité dátové štruktúry čas $O(n^2)$.
- Výpočet postupnosti konvexných obalov bude trvať maximálne $O(\frac{n^2 \log n}{3})$, lebo:
  - Vypočítanie prvého konvexného obalu trvá $O(n\log n)$.
  - Následné počítanie ďalšieho obalu bude trvať $O((n-k)\log(n-k))$, kde k je počet vrcholov predošlého obalu.
  - Obalov je v najhoršom prípade $\frac{n}{3} + 1$ (postupnosť do seba vnorených trojuholníkov), takže najhorší prípad bude trvať $\sum_{k=0}^{\frac{n}{3}} (n - 3k)\log(n - 3k)$, čo vieme ohraničiť $O(\frac{n^2 \log n}{3})$.
- Pripojenie nadradeného konvexného obalu s m vrcholmi ku dovtedy vypočítanej kružnici s k vrcholmi bude trvať maximálne $2mk^2$.
  - Pridanie jedného vrcholu bude trvať $2k^2$, čo je časová náročnosť výpočtu nárastu dĺžky kružnice pri vložení vrcholu zahŕňajúca testovanie na k možných kolízií s ostatnými hranami kružnice.

- Pridávame *m* vrcholov.

- Pridanie všetkých konvexných obalov bude trvať v najhoršom prípade (napríklad do seba vnorené trojuholníky) $6\sum_{k=1}^{\frac{n}{3}-1}(n-3k)^2$, čo vieme ohraničiť $O(n^3)$

- Takže celková časová zložitosť algoritmu bude v najhoršom prípade $O(n^3)$.

Priestorová zložitosť algoritmu bude $O(n^2)$ podobne ako v predchádzajúcom prípade, nakoľko boli použité tie isté dátové štruktúry.
Algoritmus bol testovaný na rovnakej množine grafov ako predchádzajúci.

6. Výsledky

V tejto kapitole sa venujeme popisu výsledkov testovania nami navrhovaných algoritmov a porovnaniu ich riešení s riešeniami získanými metódou kompletného prehľadávania, či NS algoritmom s optimalizáciou alebo bez nej. Popisované riešenia NS sú vždy najlepšie riešenia tejto metódy na danom grafe, získané spúšťaním algoritmu pre všetky možné štartovacie vrcholy. Avšak ak by sme chceli byť dôslední, bolo by potrebné buď brať do úvahy priemerný výsledok NS, alebo ho rozšíriť na prehľadanie všetkých možností voľby počiatočného vrcholu. Toto zväčší časovú náročnosť algoritmu o jeden rád na kubickú. Pridanie lokálnej optimalizácie časovú náročnosť rádovo nezmení. Časová náročnosť ostatných dvoch metód je približne rovnaká.

6.1 Algoritmus lineárneho prehľadávania

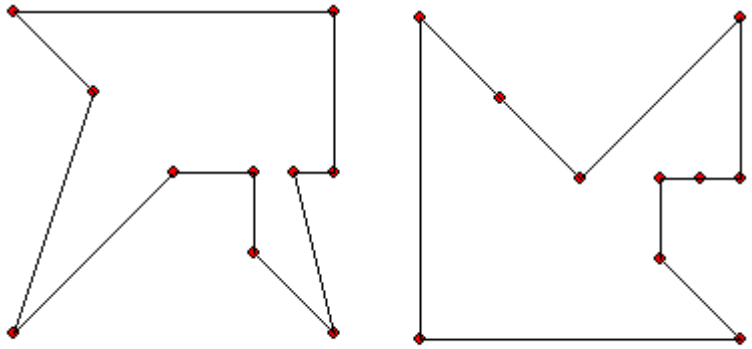
Na náhodne generovaných grafoch s menším počtom vrcholov(pre $n \leq 12$ hlavne kvôli porovnaniu s výsledkami kompletného prehľadávania) dosahuje algoritmus takmer 100 % úspešnosť. V každom prípade bol úspešnejší ako NS algoritmus hľadania najbližšieho suseda, ktorý vo väčšine prípadov generoval prekríženie hrán. Pri menšom počte vnútorných vrcholov v grafe je to zdôvodniteľné tým, že takéto prípady obsahujú minimálny počet vkladania vrcholov do už vytvorenej kružnice.
Po vypnutí lokálnej optimalizácie už rozdiely v kvalite riešení medzi NS a lineárnym prehľadávaním neboli také markantné, ale rozdiely oproti optimálnemu riešeniu narástli. V podstate algoritmus bez optimalizácie bol schopný nájsť optimálne riešenie len pri malom počte vnútorných vrcholov.

Po pridaní lokálnej optimalizácie aj do NS algoritmu, tento v najlepšom prípade nachádzal porovnateľné riešenia.

Pri grafoch s väčším počtom vrcholov (viac ako 30) sa rozdiely medzi riešeniami dosiahnutými algoritmom LP a predpokladaným optimálnym riešením zväčšili.
Keďže v aplikácii nebol implementovaný algoritmus na odhad hodnoty optimálneho riešenia, rozdiel medzi vypočítanou hodnotou a optimálnou bol odhadovaný na základe vizuálnej informácie. Pre grafy, v ktorých značne narástol aj počet vnútorných vrcholov,

dosahoval algoritmus bez optimalizácie kvalitatívne približne rovnakú úroveň riešenia ako metóda hľadania najbližšieho suseda avšak oproti nemu bol podstatne citlivejší na vzájomné usporiadanie vrcholov v rámci skúmaného grafu.

Počas testovania sa zistilo, že najväčšie nedostatky má LP algoritmus na vstupoch, ktoré obsahujú vrcholy s navzájom rovnakými vzdialenosťami, alebo na grafoch, ktoré sú určitým spôsobom symetrické. Ukážkovým príkladom takéhoto grafu je už viackrát spomínaný Dantzig graf. Pre lepšiu ilustráciu problému je v ľavej časti nasledujúceho obrázku umiestnené riešenie dané LP metódou a pre porovnanie aj výsledok získaný metódou hľadania najbližšieho suseda.



Obrázok 15 Porovnanie výsledkov metód pre Dantzig graf. Vľavo LP, vpravo NS

Jednoduché hľadanie najbližšieho suseda v tomto prípade dosiahlo lepší výsledok, a to aj bez optimalizácie, ktorá by, vďaka symetrii grafu, riešenie nezlepšila,. Pre dosiahnutie lepšieho výsledku by bolo potrebné zaviesť optimalizáciu, ktorá bude vymieňať väčšiu časť grafu. Toto opatrenie by však pri podobne symetrickom grafe s oveľa väčším počtom vrcholov nebolo dostačujúce.

Výkon a nedostatky algoritmu lineárneho prehľadávania možno zhrnúť nasledovným spôsobom:

- Na grafoch s menším počtom vrcholov s použitím optimalizácie bol algoritmus úspešný a lepší ako metóda hľadania najbližšieho suseda. Po vypnutí lokálnej optimalizácie alebo jej pridaní do druhého algoritmu sa ich výkon vyrovnal.
- Na grafoch s väčším počtom vrcholov, vzhľadom k nedostatku informácií o optimálnom riešení, sa porovnávané algoritmy javili ako rovnocenné.

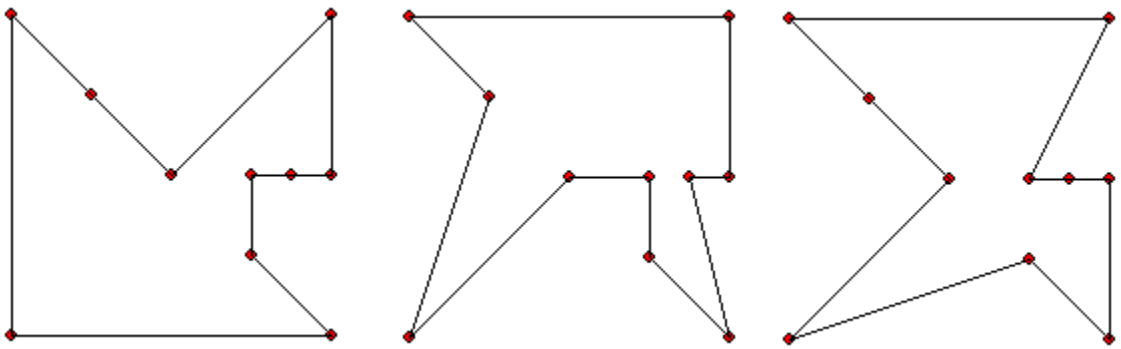
- Na Dantzig grafe bolo riešenie získané metódou hľadania najbližšieho suseda kvalitatívne lepšie než riešenie nájdené LP algoritmom.

6.2 Algoritmus postupnosti konvexných obalov

Podobne ako LP algoritmus, aj tento algoritmus bol pre grafy s menším počtom vrcholov lepší ako NS algoritmus a vo väčšine prípadov našiel optimálne riešenie. Do tohto algoritmu nebola zaradená lokálna optimalizácia.

Pre grafy s väčším počtom vrcholov už výsledky neboli také dobré. Podobne ako v predchádzajúcom prípade, aj tu sa ukázala klesajúca kvalita dosiahnutých výsledkov s rastúcim počtom vrcholov grafu. Pri porovnaní výsledkov s algoritmami LP a NS bez optimalizácie bola kvalita riešení približne rovnaká. Pri porovnaní s predošlými algoritmami so zaradenou lokálnou optimalizáciou boli riešenia, nájdené touto metódou, výrazne horšie.

Podobne ako v predchádzajúcom prípade pre symetrické grafy, mal tento algoritmus problém nájsť optimálne riešenie aj pri malom počte vrcholov. Na pripojenom obrázku sú vedľa seba zobrazené riešenia Dantzig grafu pre všetky tri porovnávané metódy v poradí NS, LP a metóda spájania konvexných obalov.



Obrázok 16 Výsledky pre Dantzig graf v poradí NS, LP, KO

Poradie na obrázku zároveň zodpovedá aj kvalite vypočítaného riešenia, nakoľko najlepší výsledok v tomto prípade vypočítal NS algoritmus aj bez optimalizácie. Dĺžky ciest boli v poradí 761, 773, 776. Dĺžka optimálnej cesty, zistená metódou kompletného prehľadávania je 756.

Pre lepšiu názornosť rozdielu v efektivite a kvalite dosahovaných riešení pre tri uvažované algoritmy, dávame čitateľovi do pozornosti nasledujúcu tabuľku, ktorá porovnáva tieto tri algoritmy a algoritmus NS s optimalizáciou na vzorke 10 grafov s 30 vrcholmi. Posledný údaj v tabuľke je priemerná hodnota.

NS	LP	CH	NS OPT
2067	1986	2062	1997
1772	1481	1551	1514
1695	1584	1729	1561
1548	1474	1500	1461
2475	2184	2504	2176
1859	1878	1902	1819
2282	1748	1913	1748
1225	1185	1409	1178
1299	1179	1412	1203
2031	1719	1927	1810
1825,3	1641,8	1790,9	1646,7

Tabuľka 1 Porovnanie výsledkov algoritmov na 10 náhodne vybraných grafoch s 30 vrcholmi

7. Diskusia

7.1 Priebeh experimentu a návrh algoritmov

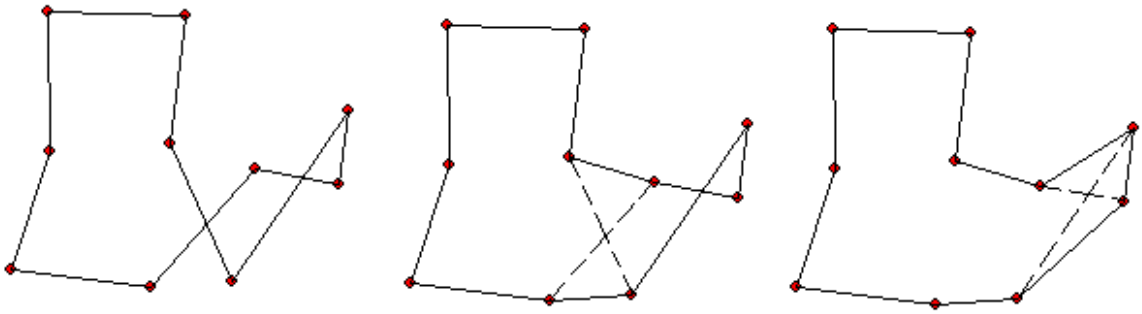
Experimentálna časť práce prebehla pomerne hladko. Aplikácia navrhnutá za týmto účelom ho plne splnila. Jediným jej nedostatkom sa ukázala byť možnosť hľadať optimálne riešenia len pre grafy s malým počtom vrcholov. Počiatočná predstava, že menší počet vrcholov v skúmaných grafoch celkom postačuje na otestovanie väčšiny „problémových“ prípadov sa ukázala nepravdivá. Niektoré nedostatky testovaných algoritmov sa prejavili až pri grafoch nad 40 vrcholov. V prípade pokračovania v práci by bolo potrebné zaradiť do aplikácie metódu na odhadovanie dolnej hranice pre optimálne riešenie. Tento účel by mohol splniť napríklad aproximatívny algoritmus, pracujúci v polynomiálnom čase.

Pri návrhu algoritmu, ktorý by mal simulovať ľudskú prácu na riešení problému, bolo najťažšou úlohou interpretovať ľudskú schopnosť držať smer postupu v grafe. Navrhovaný prístup tento zámer splnil len čiastočne. Najväčším problémom bolo interpretovať schopnosť dočasne porušiť smer pre odbočku k lokálnej časti grafu, ktorá ležala mimo hlavného smeru prehľadávania. Keďže je ťažké, vzhľadom na počet možností, odhadnúť, kedy by sa mal algoritmus zase vrátiť na cestu smerom, ktorým začal, bolo potrebné tieto odbočky vyriešiť následným pridávaním vrcholov metódou najlacnejšieho vkladu. Týmto krokom sa algoritmus vzďaľuje od riadeného postupu po grafe, ktorý je vlastný ľudskému riešeniu a vrcholy pridáva len na základe najmenšieho nárastu celkovej dĺžky cesty. Tento postup môžeme priblížiť lokálnemu prehľadávaniu tým, že pridávame tieto vrcholy postupne k hranám postupne v doteraz vytvorenej kružnici a nie globálne v celom grafe.

Návrh algoritmu spájania konvexných obalov vychádza z mierneho upustenia požiadavky na najlacnejší vklad vrcholu do grafu za účelom získania lepšieho riešenia. Nakoniec sa táto metóda ukázala ako najmenej úspešná. Je však možné predpokladať, že možnosť optimalizácie by mohla spôsobiť zmenšenie rozdielov medzi výkonmi použitých metód.

7.2 Zložitosť a výkon navrhovaných algoritmov

Čiastkové porovnanie algoritmov na malých testovacích vzorkách, uvedené v prehľadovej tabuľke v predchádzajúcej kapitole, verne odráža celkové výsledky testovania a vzájomného porovnávania efektivity jednotlivých algoritmov. Algoritmy s vyššou časovou zložitosťou mali vo všeobecnosti lepšie výsledky ako algoritmy s nižšou časovou zložitosťou, pričom najlepšie výsledky dosiahol algoritmus LP, ktorý maj aj zo všetkých použitých (okrem metódy kompletného prehľadávania) najväčšiu časovú zložitosť.
V LP algoritme mala zásadný vplyv na časovú zložitosť dĺžka počiatočnej kružnice. Táto bola vytváraná so snahou maximálne udržať smer postupu. V malých grafoch s nie veľkým počtom vnútorných vrcholov bola dĺžka tejto iniciálnej kružnice vo všeobecnosti väčšia. Následne postup obsahoval menej vkladaní bodov s najmenším pridaním k celkovej dĺžke kružnice. V prípade, keby sa podarilo navrhnúť počiatočnú cestu lepšie (ako sme už spomínali, keby lepšie odrážala lineárny postup človeka), bolo by možné vylepšiť aj priemerný prípad práce tohto algoritmu.
V KO algoritme mal zásadný vplyv na časovú zložitosť rovnako počet vkladaných vrcholov.
Tu sa dostávame k zaujímavému problému, ktorým je spôsob vkladania vrcholov bez povolenia vzájomného pretínania sa hrán. V oboch prípadoch nám tento postup spôsobil nárast časovej zložitosti algoritmov o jeden rád. Z $O\left(n^3\right)$ na $O(n^4)$ v prvom a z $O(n^2 \log n)$ na $O(n^3)$ v druhom prípade. Je potrebné spomenúť, že v prvom prípade bol tento postup motivovaný ľudskou postupom na riešení problému. Každý z riešiteľov totiž veľmi rýchlo prišiel na to, že kríženie hrán zväčšuje dĺžku kružnice a preto sa mu už dopredu pri jej tvorení vedome vyhýbal. Navyše táto vlastnosť zabezpečuje, že vrcholy konvexného obalu budú vo výslednej kružnici v poradí, čo je nevyhnutná vlastnosť optimálneho riešenia. V druhom algoritme, kde to nevyžaduje postup, by sme však mohli túto podmienku porušiť. Tým by sme znížili časovú náročnosť o jeden rád. Následne by sme mohli vzniknuté kríženia odstrániť lokálnou optimalizáciou s podľa nasledovného obrázku:



Obrázok 17 Príklad optimalizácie odstraňovaním krížiacich sa hrán

Predpokladaný časový odhad takejto optimalizácie závisí od toho, koľko krížení v priemernom prípade generuje použitý algoritmus. Pokiaľ by algoritmus generoval len $O(n)$ krížení, tak ich odstránenie bude trvať $O(n^3)$, čo nám celkový výsledok nezlepší.

Oba skúmané algoritmy boli vysoko citlivé na počet vnútorných bodov v grafe a nedokázali dostatočne efektívne pracovať v prípade symetrických grafov, čo možno hodnotiť ako ich najväčšie nedostatky. Táto skutočnosť sa dala očakávať, keďže od počtu vnútorných bodov závisí aj dĺžka prvotnej kružnice v prvom prípade, aj počet konvexných obalov v druhom prípade. A od toho závisí aj počet pridávania bodov, čo je v podstate krok, ktorý vnáša určitú „náhodnosť“ do postupu. Symetria ako taká zase pridáva počet možností ako jeden konkrétny vrchol do grafu vložiť, takže tiež v podstate zvyšuje „neistotu“. Symetria by sa dala riešiť len prehľadávaním, čím veľmi narastie časová zložitosť algoritmov. Keďže NS algoritmu počet vnútorných bodov v grafe nekomplikuje postup, tak rozdiely medzi výsledkami jednotlivých postupov sa pri väčšom počte vnútorných bodov zmenšovali, prípadne zväčšovali v neprospech testovaných algoritmov.

Ostáva otázka, či by bolo možné nejakým spôsobom skombinovať „nezávislosť“ NS algoritmu od počtu vnútorných bodov a obmedzenú „žravosť“ navrhovaných algoritmov KO a LP. Ako možnosť sa ponúka už spomínané lepšie držanie smeru postupu a umožnenie kontrolovaných lokálnych odchýlok od tohto smeru v algoritme LP. Tým by sa zväčšila jeho nezávislosť od počtu vnútorných vrcholov grafu a zároveň by sa zachovala požiadavka na „striedmosť“ greedy prístupu.

V študovanej literatúre bola vo väčšine prípadov veľmi vyzdvihovaná ľudská schopnosť riešenia vizuálne zadaného problému. Autori popisovali čas potrebný pre riešení problému človekom ako lineárny. V čom teda spočíva jeho linearita? Pri pokuse popísať výkon

človeka algoritmicky sa v podstate ukázalo, čo bolo aj očakávané. Lokálne prehľadávanie, nekrižovanie hrán, všetky tieto problémy rieši ľudský riešiteľ ľudovo povedané „od pohľadu“. Takže v konštantnom čase vyrieši niečo, čo algoritmu trvá polynomiálny čas. V tomto sa počítač človeku pravdepodobne tak skoro nevyrovná. Druhou stranou mince je ale kvalita riešenia problému. Pokus o simuláciu ľudského postupu nakoniec produkoval lepšie výsledky v porovnaní s ostatnými popisovanými algoritmami. Ale líšil sa od nich jednou podstatnou vlastnosťou, ktorou sa líšil aj ľudský postup pri riešení vo väčšine preštudovaných prác a tou je následná lokálna optimalizácia, ktorá je človeku, na rozdiel od algoritmu, pri vizuálnom riešení problému vždy umožnená. Porovnávaným algoritmom umožnená nebola. Takže v prípade, keď algoritmu povolíme následnú optimalizáciu jeho riešenia, prebiehajúcu v čase podobnom ako je čas práce algoritmu, výsledky jeho a ľudskej práce na probléme sa k sebe trochu priblížia.

7.3 Zhodnotenie použitej metódy

V závere diskusie sa dostávame späť k začiatku tejto práce a k počiatočnej motivácii. Bol ňou zámer navrhnúť algoritmus pre riešenie vybraného ťažkého problému na základe pozorovania a hodnotenia postupu ľudského riešiteľa. Ako sa podarilo tento cieľ splniť? Výsledkom je algoritmus pracujúci v čase $O(n^4)$, ktorého riešenia sa ukazujú byť v zásade lepšie, než riešenia ostatných navrhovaných a testovaných metód. V diskusii navrhujeme možné ďalšie zlepšenia, predĺženie prvej cesty, založenej na lepšom pochopení vlastností najkratšej hamilntonovskej kružnice pre daný graf a pochopení práce ľudskej intuície pri budovaní kružnice. Aj keď očakávania boli vyššie než dosiahnutý výsledok, v zásade navrhovaný algoritmus splnil aspoň úlohu mierneho vylepšenia kvality riešenia oproti bežným metódam za cenu nie príliš veľkého nárastu časovej zložitosti. Problém obchodného cestujúceho počas práce odkryl svoju zložitosť a svoje základné vlastnosti, avšak k naozajstnému pochopeniu jeho vnútorných zákonitostí (príklad o Vietových vzorcoch optimisticky použitý v úvode práce) bude potrebné oveľa dlhšie a podrobnejšie štúdium.

Čo sa týka samotného skúmania ľudského riešenia ako takého, tento postup sa zdá byť veľmi užitočný. Počas práce som študoval literatúru zaoberajúcu sa touto témou, komunikoval s priateľmi, či známymi, ktorí tiež nejakým spôsobom vedecky pracujú alebo

pracovali aj v oblasti informatiky. Diskusie s nimi mali veľký význam, ale všetky mali jednu spoločnú črtu a tou bol veľmi „informatizovaný“ pohľad na vec. Diskusia sa obrátila na oblasti ako časová zložitosť, osekávanie vetiev kompletného prehľadávania, prípadne približné metódy výpočtu. Laikovi, ktorému sa len ponúkne problém na riešenie, a ktorý nevie nič o jeho skutočnej vnútornej zložitosti, nie je nič nemožné a mnohokrát urobí intuitívne krok, na ktorý človek mysliaci pod vplyvom množstva preštudovanej literatúry tak jednoducho nepríde. A to je kladný prínos skúmania ľudského myslenia za účelom vytvorenia algoritmu pre riešenie problému.

Navrhovaná metóda ako taká sa teda ukázala byť použiteľná na čiastočné vylepšenie výsledkov pre riešenie daného problému. V prípade iných, tak isto jednoducho zadaných a bežnému človeku zrozumiteľných problémov, by mohlo byť možné takto dosiahnuť možno aj väčšie zlepšenie doterajších výsledkov vzhľadom na to, že problém obchodného cestujúceho má veľmi dlhú históriu a vysokú úroveň preskúmania. Ak sa nájde „voľné pole“ v priestore takto jednoducho zadaných zložitých problémov, navrhovaná metóda by mohla pomôcť aspoň k jeho hlbšiemu pochopeniu.

8. Záver

Výsledkom skúmania ľudského postupu pri riešení vizuálne zadaného ťažkého problému v tejto práci je algoritmus lineárneho prehľadávania. Jeho riešenia sú porovnateľné a vo väčšine prípadov o trochu lepšie, ako riešenia porovnávaných podobných metód. V jeho postupe existuje viacero možností vylepšenia, ktoré by mohli viesť k posunutiu jeho riešení smerom k optimálnemu riešeniu. V zásade je teda možné povedať, že metóda skúmania ľudského postupu by mohla viesť k dobrým výsledkom, keď už nie k absolútnemu riešeniu problému, tak aspoň k vylepšeniu doterajších výsledkov. V tomto práca splnila svoj účel.

V budúcnosti by tento postup mohol byť použitý aj pre iné problémy, pre ktoré existuje jednoduché, človeku ľahko pochopiteľné zadanie spojené s možnosťou jednoduchej a prehľadnej vizualizácie.

9. Referencie

[1] Dantzig G.B., Fulkerson D.R., and Johnson S. (1959). On a Linear Programming Combinatorial Approach to the Traveling Salesman Problem, *Operations Research* 7, p. 58-66.

[2] Diestel R., *Graph Theory*, Springer-Verlag New York 2000, ISBN 0-387-98976-5

[3] Dry M., Lee M. D., Vickers D., and Hughes P. (2006), Human Performance on Visually Presented Traveling Salesperson Problems with Varying Numbers of Nodes, *The Journal of Problem Solving*, volume 1, no. 1, p. 20-32

[4] Flood, M. M. (1956). The travelling salesman problem. *Operations Research*, 4, p. 61-75.

[5] Graham, S. M., Joshi, A., & Pizlo, Z. (2000). The traveling salesman problem: A hierarchical model. *Memory & Cognition*, 28(7), 1191-1204.

[6] Hromkovič J., *Algorithmics for hard problems (Introduction to Combinatorial optimization, Randomization, Approximation and Heuristics)*, Springer-Verlag 2002, ISBN 3-540-44134-4

[7] Krolak P., Felts W., Marble G. (1971). A man-machine approach toward solving the traveling salesman problem. *Communications of the ACM* , volume 14, Issue 5, p. 327-334, ISSN: 0001-0782.

[8] Lee, M. D., & Vickers, D. (2000). The importance of the convex hull for human performance on the Traveling Salesman Problem: A comment on MacGregor and Ormerod (1996). *Perception & Psychophysics*, 62(1), 226–228.

[9] MacGregor J. N., and Ormerod T. (1996). Human performance on the traveling salesman problem. *Perception & Psychophysics*, 58 (4), p. 527-539

[10] Michie, D., Fleming J. G., and Oldfield J. V (1968). A comparison of heuristic, interactive and unaided methods of solving a shortest route problem. *Machine intelligence*, 3, p. 245-255.

[11] Polivanova, N. I. (1974). On some functional and structural features of the visual-intuitive components of a problem-solving process. *Voprosy Psikhologii [Questions of Psychology]*, 4, p. 41-51.