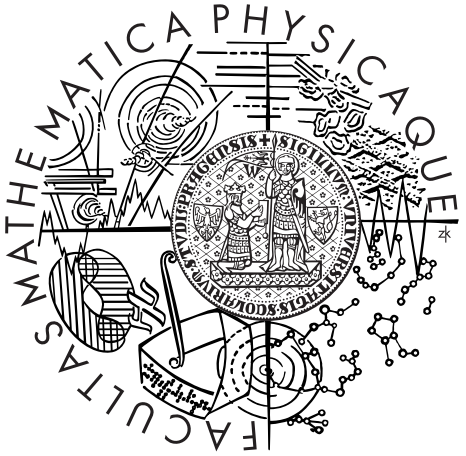
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Marek Beliš

Procházky v časově proměnlivém grafovém modelu

Katedra aplikované matematiky

Vedoucí bakalářské práce:	RNDr. Ondřej Pangrác, Ph.D.
Studijní program:	Informatika
Studijní obor:	Obecná informatika

Praha 2014

Chcel by som v prvom rade vyjadriť veľkú vďačnosť môjmu vedúcemu práce, RNDr. Ondřejovi Pangrácovi, Ph.D., za zhovievavosť a trpezlivosť pri vedení. Takisto chcem poďakovať rodičom za pochopenie môjho štúdia a všetkým mojim blízkym kamarátom za podporu pri tvorbe tejto práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborních zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Procházky v časově proměnlivém grafovém modelu

Autor: Marek Beliš

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D., Informatický ústav Univerzity Karlovy

Abstrakt: Orientační běh s migrujícími kontrolami můžeme reprezentovat grafovým modelem s ohodnocenými vrcholy a orientovanými hranami, kde ohodnocení vrcholů je proměnlivé v čase. Hledání optimální cesty představuje exponenciálně složitý problém, který je v reálnem čase řešitelný jen pomocí omezujících podmínek a heuristik. Program *SMIK solver* pomocí naimplementovaných heuristik hledá optimální cestu vícero způsoby. V práci najdeme popis problému a algoritmů, programátorskou dokumentaci s popisem metod výpočtu a porovnání výsledků programu s reálnými závody.

Klíčová slova: heuristiky, optimalizace, procházky v grafech

Title: Walks in time-dymanic graph model

Author: Marek Beliš

Department: Department of Applied Mathematics

Supervisor: RNDr. Ondřej Pangrác, Ph.D., Computer Science Institute of Charles University

Abstract: Orienteering with moving control points can be represented as a graph model with ranked vertices and oriented edges. The rank of vertex depends on the actual time. Searching for the most optimised walk through this graph is exponentially difficult problem which can be in real-time solved only with limiting conditions and using of heuristics. *SMIK solver* software provides implemented heuristics for searching the best walk in several ways. In the thesis we can find description of the problem and algorithms, programming documentation with description of used methods and comparation of results to the real competition.

Keywords: heuristics, optimization, walks in graphs

Názov práce: Prechádzky v časovo premenlivom grafovom modeli

Autor: Marek Beliš

Katedra: Katedra aplikovanej matematiky

Vedúci bakalárskej práce: RNDr. Ondřej Pangrác, Ph.D., Informatický ústav Karlovej Univerzity

Abstrakt: Orientačný beh s migrujúcimi kontrolami môžeme reprezentovať grafovým modelom s ohodnotenými vrcholmi a orientovanými hranami, kde hodnotenie vrcholov je premenlivé v čase. Hľadanie optimálnej cesty predstavuje exponenciálne zložitý problém, ktorý je v reálnom čase riešitelný len pomocou obmedzujúcich podmienok a heuristík. Program *SMIK solver* za pomoci naimplementovaných heuristík hľadá optimálnu cestu viacerými spôsobmi. V práci nájdeme popis problému a algoritmov, programátorskú dokumentáciu s popisom metód výpočtu a porovnanie výsledkov programu s reálnymi pretekmi.

Kľúčové slová: heuristiky, optimalizácia, prechádzky v grafoch

Obsah

Úvod

Orientačný beh sú preteky jednotlivcov, dvojíc či skupín, kde úlohou súťažiacich je vyzbierať čo najviac kontrol vo vymedzenej dobe. Obvykle sa konajú v prírode, hlavne v lesoch, menej časté sú preteky v urbanizovaných oblastiach. Kontroly môžu mať rôzne bodové ohodnotenie. Primárnym kritériom hodnotenia je počet nazbieraných bodov, až sekundárnym hodnotením je čas v cieli. Bežec s vyšším počtom bodov sa umiestni lepšie ako bežci, ktorí dobehli do cieľa skôr, ale s menším počtom nazbieraných bodov. Za prekročenie časového limitu pretekov sa súťažiacim strhávajú body. Základnou pomôckou je podrobná mapa oblasti s vyznačenými kontrolami a často aj buzola.

V základnej verzii orientačného behu súťažiaci obdrží mapu 5 minút pred štartom. Preteky jednotlivcov trvajú 60 až 120 minút a za prekročenie časového limitu sú progresívne bodové tresty. Za meškanie do 1 minúty sa strháva 1 bod, za meškanie viac ako 1 minúta, ale do 2 minút sa strhávajú 1 + 2 body a podobne za väčšie meškania. Preteky obsahujú 15 až 25 rovnako ohodnotených kontrol, nezávisí na poradí ich vyzbierania. Štart môže byť jednotlivo v intervaloch, v intervaloch po skupinách alebo hromadne.

Existujú variácie orientačného behu. Rádioorientačný beh, kde bežec musí najprv pomocou detektoru lokalizovať kontroly, cykloorientačné preteky, v zimnom období preteky na bežeckých lyžiach.

Medzi špecifické variácie patrí orientačný beh s migrujúcimi kontrolami, alebo skrátene *SMIK*. Rovnakou charakteristikou je 60-minútová dĺžka trvania, progresívne tresty za prekročenie časového limitu a nezávislé poradie zbierania kontrol. Od klasických pretekov sa odlišuje najmä rôznou polohou tej istej kontroly v priebehu pretekov a rôznymi ohodnoteniami kontrol. Žiadna kontrola nemôže byť v tom istom okamihu na viacerých stanovištiach. 5 minút pred pretekmi dostáva účastník okrem mapy aj zoznam stanovíšť pre jednotlivé kontroly. Na zozname sú vypísané časové rozmedzia, v ktorých sa daná kontrola nachádza na jednotlivých stanovištiach. Počet bodov za kontrolu je rovnaký v každom stanovišti. Kvôli pohybu kontrol je nutný hromadný štart [5].

Matematicky môžeme mapu pretekov reprezentovať ako orientovaný graf s váženými hranami a ohodnotením vrcholov, ktoré sa však mení v čase. Hrany reprezentujú časovú vzdialenosť medzi vrcholmi, hodnota vrcholu značí počet bodov za danú kontrolu. Aj bez premenlivého hodnotenia vrcholov je hľadanie ideálneho riešenia NP-úplné, podobné problému obchodného cestujúceho. Na rozdiel od tohoto problému pri orientačnom behu nie sú relevantné všetky hrany a nemá

zmysel zadávať úplný graf. Je vhodné vybrať susedov v krátkej časovej dostupnosti, cez ktorých sa môžeme posunúť v grafe k ďalším vrcholom [6].

Programom *SMIK solver* ponúkame užívateľovi možnosť optimalizácie jeho behu na základe zadaných vrcholov a očakávaných časov presunu medzi jednotlivými stanovišťami. Program používa heuristiky, z ktorých si užívateľ môže vybrať spôsob výpočtu. Počítanie *podľa časového okienka* dynamicky buduje cestu na základe kratších úsekov, ktoré počíta podľa veľkosti časového okienka zadaného užívateľom. Obecne platí, že s väčším časovým okienkom stúpa pravdepodobnosť presnejších a lepších výsledkov. Na druhej strane prudko rastie doba výpočtu. Pri počítaní *podľa vybraných kontrol* sa program snaží detekovať najlepšie ohodnotené kontroly a vybrať ideálnu cestu tak, aby bežec vyzbieral tieto kontroly, dobehol v limite, a zároveň pozbieral čo najviac bodov pri presune medzi danými vybranými kontrolami. Ideálny beh výpočtu je menej ako 5 minút, čo je porovnatelný čas s prípravou pred pretekmi.

1. Grafový model a algoritmy výpočtu

Problém obchodného cestujúceho, v základnej verzii alebo s obmedzenou prítomnosťou zákazníkov v mestách, je často riešeným NP-úplným problémom. V používaných algoritmoch je nutné využitie heuristík a orezávaní.

V súčasnosti existujú algoritmy, ktoré v grafoch s definovanou metrikou zaručujú 3-aproximáciu [1] a najnovšie $(2 + \epsilon)$ aproximáciu. Tieto aproximácie sa vzťahujú len na problém bez maximálneho povoleného času v neorientovaných grafoch. V prípade orientovaných grafov je možné dosiahnuť $O(\log^2 \text{OPT})$ aproximáciu, kde OPT je počet vrcholov navštívených optimálnym riešením. Pri orientovaných grafoch s časovo obmedzenou dostupnosťou vrcholov sú algoritmy s aproximáciou $O(\log^4 \text{OPT})$ [2]. Pomocou hladového algoritmu môžeme aj v týchto problémoch dosiahnuť $O(\log \text{OPT})$ [3].

K dosiahnutiu týchto hodnôt sa využívajú rôzne metódy. Častou technikou je segmentácia grafu na menšie podgrafy a ich následné spájanie [2]. Jednou z ďalších možností je budovanie stromu od štartu postupným pridávaním vyhovujúcich vrcholov ako listy stromu [4], kde sa výsledky porovnávajú s postupom vkladaním vrcholu do už vytvorenej cesty. Ďalším variantom riešenia tohoto problému sú genetické algoritmy [7].

V programe *SMIK solver* nájdeme jednu grafovú reprezentáciu a dva základné algoritmy na výpočet najlepšej cesty. Navrhnuté algoritmy predstavujú spolu s obmedzujúcimi podmienkami alternatívu k problému obchodného cestujúceho v orientovaných grafoch s časovo obmedzenou dostupnosťou vrcholov. Hlavným rozdielom oproti väčšine študovaných modelov a algoritmov je možnost navštívenia ekvivalentného vrcholu a vyzbieranie profitu na rôznych miestach v odlišných časoch.

1.1 Grafový model

Definícia. *$G = (V, \vec{E})$ je grafovým modelom orientačného behu SMIK, pokiaľ $\exists n_1$, n_2 prirodzené a*

- *$V = \{v_1, v_2, ..., v_n\}$ je množina vrcholov,*
- *$\vec{E} = \{(v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\}$ množina orientovaných hrán,*
- *$t_{\max}$ je dĺžka trvania pretekov*

- $o : V \to \{0, 1, ..., t_{\max}\}$ *priradzuje časy otvorenia vrcholu,*
- $z : V \to \{0, 1, ..., t_{\max}\}$ *priradzuje časy uzatvorenia vrcholu,*
- $p : V \to \{0, 1, ..., n_1\}$ *priradzuje bodové ohodnotenie vrcholov,*
- $k : V \to \{0, 1, ..., n_2\}$ *priradzuje čísla kontroly,*
- $l : \vec{E} \to \{0, 1, ..., t_{\max}\}$ *určuje váhu hrany*
- $s, c \in V$ *sú vrcholy reprezentujúce štart a cieľ.*

Takto zadefinovaný formálny zápis pokrýva mapu pretekov. Každý vrchol je unikátne kódovaný číslom kontroly a poradovým číslom stanovišťa. Štart a cieľ majú formálne číslo kontroly 0, rovnako aj vložené pomocné body na významných miestach či križovatkách v oblasti. Ku každému vrcholu prináleží jeho čas otvorenia a zatvorenia (pre štart, cieľ a križovatky to je dané dobou trvania pretekov) a počet bodov za kontrolu v danom vrchole. V prípade, že v čase t navštívenia vrcholu v platí $t < o(v)$ alebo $z(v) < t$, tak $p(v) = 0$. Váha hrany medzi dvomi vrcholmi je zadefinovaná ako doba presunu medzi vrcholmi, nemusí byť symetrická. Optimálnym riešením problému by v tom prípade bola cesta $P = (s, e_1, v_1, ..., e_k, c)$ taká, že $\Sigma_{e_i \in P} l(e_i) \leq t_{\max}$ a pre cestu $R \neq P$ platí, že cestou R vyzbierame menej bodov alebo prídeme do cieľa neskôr.

Ekvivalentné vrcholy sú vrcholy s priradeným rovnakým nenulovým číslom kontroly, vyzbieranie bodov na jednom z nich znemožnuje vyzbieranie bodov na ostatných vrcholoch z tejto množiny. Možno ich definovať ako rozkladové triedy podľa funkcie k. Všetky vrcholy v jednej rozkladovej triede majú rovnaký počet bodov, teda ohodnotenie bodovacou funkciou p.

Pozorovanie 1. *SMIK je blízkym problémom obchodného cestujúceho.*

Dôkaz. Riešením problému obchodného cestujúceho je nájdenie hamiltoniánskej kružnice v úplnom grafe. Jeho variáciou je štart odlišný od cieľa. V prípade *SMIK* môžeme rovnako zadať úplný graf K_n s priradením váh pre každú hranu a hľadat maximálnu cestu splňujúcu časový limit bez opakovania vrcholov na tejto ceste. □

Pre viaceré špecifiká nemôžeme v našom modeli použiť popisované riešenia z iných literatúr. Nesymetrickosť grafu je daná rôznymi dĺžkami presunu v teréne, kde cesta do kopca z vrcholu v_i do v_j je výrazne náročnejšia ako opačná cesta z kopca, $l(v_i, v_j) > l(v_j, v_i)$ a preto odpadajú riešenia založené na metrikách.

Ďalšou špecifickosťou je viacero ekvivalentných vrcholov, ktoré sa obvykle v iných riešeniach nevyskytujú.

Základnou optimalizáciou nášho modelu je, že nepoužívame úplný graf, oproti tomu ale sú povolené viacnásobné návštevy vrcholov. Pri pretekoch je vysoko nepravdepodobné presúvanie cez väčšie vzdialenosti na mape bez využívania vrcholov ležiacich na ceste či blízko cesty medzi danými vrcholmi. Preto majú vrcholy hrany len s najbližšími vrcholmi v rôznych smeroch. Presun ku vzdialenejším vrcholom sa vykonáva postupne cez susedov. Je povolené navštíviť vrcholy aj mimo ich času vhodného na zbieranie bodov či po vyzbieraní bodov na ekvivalentnom vrchole. Vtedy sa daný vrchol používa ako križovatka s nulovým ohodnotením.

Pre zjednodušenie problému v algoritmoch nie je povolený presah cez časový limit a teda počítanie penalizácií. Dôvodom je najmä budovanie ciest aj od cieľa, ktoré by bolo komplikované rôznymi časmi. Takto sa cesty od cieľa počítajú od $t_{\max}$.

Veta 1. *Nech n je počet vrcholov v grafovom modeli SMIK a m je priemerný počet susedov každého vrcholu. Časová zložitosť priemerného prípadu po uplatnení obmedzujúcich podmienok je*

$$O(m^n).$$

Dôkaz. Bez uplatnenia ďalších heuristík je najlepším algoritmom prehľadávanie do hĺbky, ktoré má exponenciálnu zložitosť v závislosti na počte vrcholov a počte susedov každého vrcholu. Aproximácia priemerným počtom susedov je možná vďaka amortizácii pri predpoklade rovnomerne rozloženého grafu. □

V programe je graf reprezentovaný pomocou zoznamu usporiadaných dvojíc pozostávajúcich z kľúča vrcholu a dátovej jednotky vrcholu samotného. Kľúč je tvorený kombináciou čísla kontroly a stanovišta, vrchol u seba uchováva všetky informácie náležiacie k priraďovacím funkciám (časy, body, číslo kontroly a stanovišťa) a zoznam susedov. Zoznam susedov je opäť usporiadaný zoznam dvojíc pozostávajúci z kľúča suseda a váhy hrany. Doplnkom je zoznam vrcholov, z ktorých existuje hrana do daného vrcholu, ktorý sa požíva pri výpočte ciest od cieľa.

1.2 Algoritmus výpočtu podľa časového okienka

Tento algoritmus kombinuje prehľadávanie do hĺbky s dynamickým programovaním. Základným parametrom algoritmu je veľkosť časového okienka. Algoritmus sa skladá z troch základných častí:

- budovanie cesty od štartu až do vyčerpania limitu stanoveného ako dve tretiny $t_{\max}$, alebo do vyzbierania všetkých kontrol. Na dĺžku časového okienka sa nájde najlepšia cesta, ale vykoná sa iba krok na prvý vrchol na nej, následne cyklus beží odznova. Motiváciou z reálnych pretekov je, že bežec lepšie odhadne svoje možnosti a kondíciu v úvode pretekov, zároveň vyzbieranie všetkých kontrol najrýchlejšími súťažiacimi je blízke času trvania pretekov $t_{\max}$;

- v prípade, že prvá časť nevyzbiera všetky kontroly, spustí sa striedavé budovanie cesty. Pokračuje sa v budovaní doterajšej cesty a začne sa budovať cesta od cieľa v čase ukončenia pretekov. Opäť v porovnaní s pretekmi je v tomto čase nutné plánovať cestu do cieľa a je zbytočné zbierať kontroly, ktoré sú v blízkom okolí cieľa;

- po poklese zvyšného času medzi jednotlivými cestami pod hodnotu časového okienka sa algoritmom klasického prehľadávania do hĺbky prepoja tieto dve cesty.

1.2.1 Popis algoritmu

Pre lepší popis algoritmu zadefinujme v_v ako vrchol, ktorým končí doteraz vypočítaná najlepšia cesta. Ďalej potrebujeme v_a ako aktuálny vrchol, na ktorom algoritmus práve počíta, t_o zostávajúci čas z časového okienka, t_a čas, ktorý uplynul od začiatku pretekov a b ako vyzbierané body pri výpočte cesty.

Algoritmus začína korektným priradením $v_v = s$, $t_a = 0$, $v_a = v_v$. Následne sa rekurzívne nájde najlepšia cesta v okolí v_a tak, aby platilo $t_o > 0$ a neprekračoval sa limit okienka. Pre každého suseda v_s vrcholu v_a, ktorý spĺňa podmienku $l(v_a, v_s) < t_o$ sa pustí prehľadávanie do hĺbky s parametrami $(t_a = t_a + l(v_a, v_s); t_o = t_o - l(v_a, v_s); v_a = v_s; b)$.

Na úvod rekurzívnej funkcie sa testuje prítomnosť kontroly na stanovišti

$$o(v_a) \leq t_a \leq z(v_a) \Rightarrow b = b + p(v_a).$$

a zvýši sa počet bodov. V prípade, že pre každého suseda v_s vrcholu v_a platí, že $t_o < l(v_a, v_s)$ alebo už nie sú ďalší nenavštívení susedia, nastáva porovnanie dosiahnutých výsledkov s aktuálne najlepšími. Ak sme počet získaných bodov vylepšili alebo dosiahli rovnaký počet bodov za kratší čas, tieto hodnoty dosiahnutých bodov a času sa uložia. Neukladá sa celá obsiahnutá cesta.

Po vynorení rekurzie k vrcholu v_v algoritmus overí, či sa výsledky zmenili od zanorenia cez vrchol v_s. Ak áno, a teda sme ich vylepšili, vrchol v_s sa uloží

ako následník v_{suc} vrcholu v_v. Po prejdení všetkých susedov vrcholu v_v algoritmus vykoná krok na vrchol v_{suc}.

Krokom na v_{suc} je potrebné spustiť výpočet znova s novými parametrami $(v_v = v_{suc}; t_a = t_a + l(v_v, v_{suc}))$. Začína nový krok cyklu prvej časti algoritmu, kde sa v úvode k vrcholu uloží t_a ako čas návštevy a overuje sa, či je daný vrchol otvorený. Ak platí $o(v_v) < t_a < z(v_v)$ a zároveň $p(v_v) > 0$, tak k dosiahnutým bodom na tejto už nemennej časti cesty sa pridá hodnota $p(v_v)$. Pre ďalší výpočet je potrebné vyradiť ohodnotenie ekvivalentných vrcholov, ktoré sa formálne vykoná priradením

$$\forall v \in V : k(v) = k(v_v) \Rightarrow p(v) = 0.$$

Cyklus pokračuje hľadaním optimálnej cesty od vrcholu v_v s maximálnou hodnotou časového okienka. Prvá časť algoritmu je ukončená podmienkou $t_a > \frac{2}{3}t_{\max}$, ktorá vedie k druhej časti algoritmu, alebo $\forall v \in V : p(v) = 0$, ktorá vedie priamo na tretiu časť.

V druhej hlavnej časti algoritmu sa metóda budovania cesty z prvej časti strieda s ekvivalentnou metódou na budovanie cesty od cieľa. Definuje sa vrchol v_k ako prvý vrchol na najlepšej ceste, ktorá končí v c a $t_{a,k}$ ako čas od štartu, v ktorom je daný vrchol navštívený. Na úvod sa priradí $v_k = c$, $t_{a,k} = t_{\max}$ a k výpočtu sa používa analogická metóda s prvou časťou algoritmu s malou zmenou – využívajú sa zoznamy vrcholov, pre ktoré je v_k susedom. Táto druhá časť algoritmu je ukončená podmienkou $t_{a,k} - t_a < t_o$, teda že rozdiel medzi časmi v ceste od štartu a od cieľa je menší ako základný parameter algoritmu – časové okienko.

V poslednej hlavnej časti algoritmu sa základným prehľadávaním do hĺbky hľadá cesta medzi v_v a v_k, ktorá počas výpočtu spĺňa podmienku $t_a \leq t_{a,k}$. Ak takáto cesta neexistuje, z vypočítaných ciest sa odstráni menej bodovaný vrchol, prípadne vrchol z dlhšej cesty. Ako v_v sa určí jeho predchodca alebo ako v_k sa určí jeho nasledovník. Prehľadávanie do hĺbky sa spustí znovu. Odstraňovanie vrcholov prebieha až kým neexistuje cesta medzi v_v a v_k, ktorá spĺňa podmienku času.

1.2.2 Vlastnosti algoritmu

K základným charakteristikám programu patria časové zložitosti, rozbor korektnosti a kritických miest programu.

Veta 2. *Nech n je počet vrcholov grafu, n' počet vrcholov na ceste spočítanej treťou časťou algoritmu, t_o veľkosť časového okienka. Nech m je priemerný počet*

susedov, d priemerná dĺžka hrany a r priemerný počet zanorení definované ako

$$m = \frac{|\vec{E}|}{|V|},\ \vec{E'} = \{e|e \in \vec{E}, l(e) \leq t_o\},\ d = \frac{\Sigma_{e \in \vec{E'}} l(e)}{|\vec{E'}|},\ r = \frac{t_o}{d},$$

potom priemerná časová zložitosť algoritmu s časovým okienkom je

$$O(n \cdot m^r + m^{n'}).$$

Dôkaz. Časová zložitosť algoritmu je zložená z dvoch častí a závisí na grafe a priebehu výpočtu. HĺBka rekurzie algoritmu je v každom kroku úvodných dvoch častí obmedzená časovým okienkom. Až na konštantu sa tento výpočet vykoná na každom vrchole. Niektoré vrcholy algoritmus navštívi viackrát, niektoré ani raz. U každého kroku algoritmu sa vykonáva konštantný počet operácíi, pracuje sa so štruktúrami poskytujúcimi dáta v konštantnom čase (2.2).

V prípade ideálneho výpočtu platí, ze existuje cesta kratšia ako časové okienko medzi získanou cestou od štartu a od cieľa, a teda zložitosť záverečného prehľadávania do hĺbky je oveľa menšia ako výpočet cesty. Ak táto cesta neexistuje, postupným odoberaním vrcholov sa zvyšuje časová kapacita na prehľadávanie do dĺžky, ktorá znamená nárast doby výpočtu a tento člen začína dominovať. □

Veta 3. *Nech n je počet vrcholov grafu, t_o časové okienko algoritmu, m počet susedov vrcholu s najvyšším počtom susedov. Nech d je dĺžka najkratšej nenulovej hrany alebo nulovej slučky, r je počet rekurzívnych zanorení definované ako*

$$d = \min(\{l(e)|e \in \vec{E}, l(e) > 0\} \cup \{\Sigma_{e \in C_i} l(e)|C_i \subseteq G\}),\ r = \frac{t_o}{d}$$

potom časovou zložitosťou v najhoršom prípade je

$$O(n \cdot m^r + m^n).$$

Dôkaz. Rovnako ako v priemernom prípade časovú náročnosť ovplyvňuje rekurzívna výpočtová časť a záverečné prepájanie ciest. V najhoršom prípade výpočtu budú vrcholy pribúdať po najmenších hranách, prípadne sa výpočet bude držať v hustej oblasti grafu s krátkymi hranami, čo beh výrazne spomalí.

Ďalším scenárom je, ak okolo štartu v dostupnosti časového okienka nie sú žiadne hodnotené kontroly. Program vykonáva náhodné kroky. Tieto náhodné kroky môžu vyústiť v pohyb v tesnej blízkošti štartu až do vyčerpania časového limitu. Nasleduje prehľadávanie do hĺbky cez celú mapu.

Najhoršou možnosťou je, ak graf obsahuje podgraf kružnicu s nulovou váhou. Po presunutí výpočtu na niektorý z vrcholov kružnice je algoritmus neukončitelný (z technického hľadiska program padne na preplnenie zásobníku). □

Rozobraním jednotlivých krajných možností z vety 2., vety 3. a pozorovaniami zistíme:

- zväčšovaním časového okienka t_o zvyšujeme exponent zložitosti, teda aj doba výpočtov rastie exponenciálne. Výhodou väčšieho okienka je presnejší výpočet najlepšej cesty, kedže algoritmus dovidí na viacej vrcholov;
- nezávislé nulové hrany algoritmu nevadia, v niektorých prípadoch sú dokonca nevyhnutné (ak je na jednom mieste viacero stanovíšť rôznych kontrol). Problémom sú cykly s nulovým súčtom váh hrán. Matematicky je v tom prípade zložitosť nekonečná, beh programu sa neukončí. Návod na korektné združovanie vrcholov je popísaný v prílohe;
- nemá veľký zmysel zadávať hrany s váhou väčšou ako je časové okienko. Algoritmus ich až na záverečné spájanie ciest nebude môcť použiť;
- algoritmus nemá žiadne garantované výsledky.

Najkritickejšou časťou algoritmu je vo veľa prípadoch jeho tretia časť, prepájanie vybudovaných ciest. Problém spôsobuje hlavne odlišná orientácia cesty od štartu a od cieľa, teda ak sa konce ciest k sebe nepribližujú aj napriek odoberaniu vrcholov.

Medzi vlastnosti algoritmu patrí, že vyžaduje prítomnosť štartu a cieľa v jednej komponente súvislosti a aby medzi nimi existovala cesta spĺňajúca limit.

V algoritme sme naimplementovali aj náhodný krok. Tento sa používa v prípade, že pri budovaní cesty algoritmus nenájde cestu obsahujúcu bodovanú kontrolu. Ak nastane táto sitúcia — stáva sa to najmä v závere výpočtov alebo pri príliš malom okienku — algoritmus náhodne vyberie suseda, na ktorého sa posunie. Pre túto vlastnosť môžu byť výsledky na tých istých dátach mierne odlišné. Paradoxne po dlhších výpočtoch, kedy algoritmus nevidí jasnú cestu, je vhodné výpočet niekoľkokrát zopakovať. Je veľká pravdepodobnosť, že náhodné kroky posunú výpočet do inej oblasti.

Pozorovanie 2. *Ak v okolí bodu v v dosahu časového okienka existuju bodované cesty, algoritmus vyberie najvýhodnejšiu z nich.*

Dôkaz. To plynie priamo z definície postupu algoritmu. Ak existuje bodovaná cesta, algoritmus nemusí vykonávať náhodný krok na niektorého zo susedov. Ako vrchol pre postup sa udáva vrchol, cez ktorý existuje cesta s maximálnym počtom bodov alebo tieto body vyzbiera v kratšom časovom intervale ako iná bodovo ekvivalentná cesta. □

V algoritme nie sú implementované oddychy, vyčkávanie na otvorenie kontroly. Ak algoritmus spraví na ceste slučku, na ktorej nezbiera žiadne body (až na náhodný pohyb), jedná sa o posun v čase.

Pozorovanie 3. *Ak existuje bodovaná cesta v okolí bodu v_a a algoritmus vykoná cestu $v_a v_b v_a$ bez zbierania bodov v bode v_b, ide o čakanie na otvorenie kontroly.*

Dôkaz. Pozorovanie môžeme jednoducho overiť sporom. Nech algoritmus vykoná krok z vrcholu v_a do vrcholu v_b, vráti sa do v_a a pokračuje do vrcholu v_c. Nech existuje v momente posunu do vrcholu v_b lepšia cesta cez vrchol v_c. To by ale bolo v spore s predošlým pozorovaním. Cestou cez vrchol v_b teda pozbierame viac bodov. Do vrcholu v_a sa vrátime v čase $t' = t + l(v_a, v_b) + l(v_b, v_a)$, čo vytvára nové podmienky pre výpočet posunutím sa v čase. □

1.3 Algoritmus výpočtu podľa najlepšie ohodnotených kontrol

Podstatou tohoto algoritmu je segmentácia grafu na menšie podgrafy, kde segmenty sú ohraničené kontrolami s najvyšším bodovým ohodnotením. Cesta v jednotlivých segmentoch sa vyhľadáva rekurzívne prehľadávaním do hĺbky.

Algoritmus má dve základné časti:

- detekciu najhodnotnejších kontrol a vytvorenie ich vzájomných kombinácií tak, aby každá kontrola bola v kombinácii reprezentovaná iba jedným stanovišťom. Inšpiráciou z reálneho prostredia je fakt, že bez najdôležitejších kontrol je takmer nemožné vyhrať preteky. Preto je vhodné si naplánovať cestu cez tieto kontroly.
- samotné hľadanie najlepšej cesty v každej kombinácii. V porovnaní s pretekmi to je snaha o využitie rezervného času pri presune medzi hlavnými kontrolami.

1.3.1 Popis algoritmu

Algoritmus v prvej časti zisťuje najvyššie bodové ohodnotenie $p_{\max}$ v grafe ako $p_{\max} = \max\{p(v)|v \in V\}$. Prehľadávaním grafu vytvorí množinu vybraných vrcholov $\{v|v \in V, p(v) = p_{\max}\}$. Túto množinu vrcholov je potrebné rozdeliť podla čísiel kontrol na skupiny ekvivalentných vrcholov, teda rozkladové triedy podľa čísla kontroly – rôzne stanovištia jednej kontroly. Stanovíšť je pre každú

kontrolu rovnaký počet. Vytvorením kombinácií tak, aby z každej kontroly obsahovala kombinácia práve jedno stanovište (jeden prvok z každej rozkladovej triedy podľa čísla kontroly), končí prvá časť algoritmu.

V druhej časti algoritmu je spustený cyklus cez všetky pripravené kombinácie. V jednom kroku cyklu sa spracuje jedna kombinácia. Definujme x ako počet vrcholov v jednej kombinácii. Kombinácia vrcholov $V' = \{v_1, ..., v_x\}$ je usporiadaná podľa časov otvorenia tak, že platí

$$\forall v_i, v_j \in V' : i < j \Rightarrow o(v_i) \leq o(v_j).$$

Zoradená kombinácia je presunutá do zoznamu, na úvod zoznamu je pridaný vrchol s a na záver vrchol c.

K ďalšej časti si opäť definujeme t_a ako aktuálny čas pri pohybe v grafe, v_v ako vrchol od ktorého aktuálne počítame cestu. Priradzujeme $v_v = s$ a $t_a = 0$. Algoritmus sa snaží obmedzenou rekurziou nájsť cestu medzi dvojicou vrcholov v_v a v_{v+1} tak, aby po príchode na v_{v+1} platilo $o(v_{v+1}) < t_a < z(v_{v+1})$ a aby na tejto ceste nazbieral body. Ak takáto cesta existuje, priradí sa $v_v = v_{v+1}$ a začína sa budovanie cesty medzi ďalšou dvojicou od aktuálneho času t_a. Ak neexistuje cesta, pomocou ktorej dorazíme do v_{v+1} v čase jeho otvorenia, tento vrchol sa vynechá zo zoznamu priradením $v_{v+1} = v_{v+2}$ a algoritmus sa snaží nájsť cestu medzi novou dvojicou vrcholov. Táto časť je ukončená ak $v_v = c$.

Ak existuje cesta z s do c pomocou tejto kombinácie vrcholov, či už s využitím všetkých kontrol v zozname alebo s vynechávaním, jej bodová hodnota je nenulová. Získané body a cieľový čas t_a sa porovnajú s doterajším maximom. V prípade vylepšenia cesty sa jej hodnoty stanú novým maximom, cesta sa uloží a algoritmus pokračuje ďalším krokom cyklu až do vyčerpania všetkých kombinácií.

1.3.2 Vlastnosti algoritmu

Základným a pre výpočty dôležitým obmedzením algoritmu je naimplementovaná maximálna hĺbka rekurzie pri hľadaní cesty medzi po sebe nasledujúcou dvojicou vybraných vrcholov. Experimentálne bol maximálny počet zanorení stanovený na 8. Dôvodom tohoto obmedzenia je doba výpočtu algoritmu.

Veta 4. *Nech x je počet vybraných kontrol, u počet stanovíšt jednej kontroly, m priemerný (najvyšší) počet susedov vrcholu. Potom časová zložitosť algoritmu výpočtu podľa vybraných kontrol je v priemernom (najhoršom) prípade*

$$O(u^x \cdot (x+1) \cdot m^8).$$

Dôkaz. Čas na vytvorenie všetkých kombinácií je zanedbateľný v porovnaní s výpočtami. Člen u^x predstavuje počet všetkých kombinácií a teda krokov základného cyklu. Každá kombinácia má po pridaní štartu a cieľa $x + 2$ prvkov, teda $(x + 1)$ dvojíc medzi ktorými treba vyhľadať cestu. Medzi každou dvojicou beží rekurzívne prehľadávanie o maximálnej hĺbke 8. Ostatné operácie prebiehajú v konštantnom čase. □

Pozorovaniami zistíme:

- z časovej zložitosti je zretelné, že kľúčovými parametrami sú počet vybraných kontrol a počet stanovíšť každej kontroly. V grafoch reprezentujúcich *SMIK* počet susedov zriedka býva vyšší ako 10, ale pri hustých grafoch by aj tento člen výrazne zvyšoval dobu výpočtu;
- maximálny počet navštívených vrcholov je $(x + 1) \cdot 8$. Pri malom podieli vybraných kontrol voči celkovému počtu kontrol algoritmus nepokryje celý graf. Zbytočne vysoký podiel vybraných kontrol nevedie k lepším výsledkom, algoritmus často prechádza bodmi s nulovou hodnotou;
- na rozdiel od algoritmu s časovým okienkom v tomto algoritme nevadia kružnice s váhou 0 alebo výskyt štartu a cieľa v rozličných komponentách súvislosti. Obmedzená hĺbka rekurzie zabezpečí, že výpočet bude ukončený. Ak bol zadaný nesúvislý graf, kde neexistuje cesta medzi štartom a cieľom, algoritmus nevráti žiadne výsledky.

V priebehu algoritmu ak nie je dostupná nasledujúca kontrola na 8 krokov, táto kontrola sa vynecháva.

Pozorovanie 4. *Vynechávanie nedostupných kontrol je nahradené v inej kombinácii.*

Dôkaz. Kedže program prechádza všetky kombinácie, po vynechaní kontroly v_o z kombinácie V' existuje kombinácia $V'' = V' \setminus \{v_o\} \cup \{v'_o\}$, kde v_o a v'_o sú ekvivalentné vrcholy, rôzne stanovišťia kontroly. Pre každú ďalšiu vynechanú kontrolu z V' platí, že existuje kombinácia s pôvodnými vrcholmi a ekvivalentnou náhradou tejto kontroly, teda nie je potrebné prepočítavanie predošlých výsledkov. □

2. Program SMIK solver

SMIK solver je program na hľadanie optimálnej cesty v grafe. Umožňuje manúalne zadanie dát, zmenu či opravu zadaných dát, zadanie pomocou grafického režimu, načítanie dát zo súboru, výpočet optimálnej cesty a zobrazenie výsledkov či ich uloženie do výstupného súboru. Program sme tvorili v jazyku C# s platformou .NET 3.5 pomocou programu Microsoft Visual Studio 2008 Express Service Pack 1 a je určený pre operačné systémy Windows. Jeho základné časti sú zobrazovacia trieda *UserInterface*, výpočtová trieda *Solver*, dátové triedy *Vrchol*, *Vysledok*, *Suradnice* a súbor *ResourcesSK.resx*. Ostatné súbory sú generované programom *Visual Studio* automaticky na základe dizajnu programu.

2.1 Triedy Vrchol, Suradnice a Vysledok

Tieto dátové triedy používame na reprezentáciu vrcholov v grafovom modeli a reprezentáciu nájdenej cesty.

Trieda *Vrchol* obsahuje premenné na uchovanie informácií ziadaných grafovým modelom (1.1). Zoznam susedov je reprezentovaný pomocou slovníka, *Dictionary<string, int>*, ktorý používá kľuče vrcholov v spojení s dĺžkou presunu na daného suseda, rovnako je reprezentovaný aj zoznam spätných susedov. Kľúče susedov sú uložené v zozname, *List<string>*. Okrem týchto premenných trieda obsahuje ešte konštruktor a metódu *PridajSuseda*, ktorá na danom vrchole pridá do zoznamu informácie o susedovi podľa zadaných parametrov.

Suradnice je malou dátovou triedou s dvomi premennými a konštruktorom, ktorá uľahčuje uchovávanie informácii o pozícii vrcholu v grafickej mape.

Trieda *Vysledok* sa používa pri reprezentovaní optimálnej cesty. Okrem konštruktoru obsahuje premennú typu *Vrchol* na uchovanie údajov o vrchole, číselnú premennú na čas prechodu daným vrcholom a logickú premennú, ktorá vyjadruje časový stres pri prechode vrcholom.

2.2 Trieda Solver

Solver je výpočetnou triedou, ktorá od *UserInterface* preberá dáta a tvorí štruktúru pre výpočet najlepšej cesty. Samotný výpočet prebieha priamo v tejto triede bez zasahovania či využívania metód *UserInterface*.

Základom tejto triedy je slovník *Dictionary<string, Vrchol> vrcholy*, ktorý reprezentuje zadaný graf (1.1). Trieda *Dictionary* reťazí kolízie kľúčov do zozna-

mov, kde pre prístup platí časová zložitosť

$$O(\frac{n}{m}),$$

kde n je počet prvkov a m je počet priehradiek. Táto štruktúra ale zároveň udržuje počet priehradiek m zhodný s počtom všetkých zadaných prvkov n, kde vďaka amortizácii platí

$$m = n \Rightarrow O(\frac{n}{m}) = O(1),$$

čím zabezpečuje rýchly prístup k dátam pomocou unikátneho kľúča v konštantnom čase [8].

Trieda obsahuje aj:

- obecné premenné na uchovanie potrebných dát pre výpočet, okrem iných hlavne dĺžku trvania pretekov, dĺžke časového okienka, počte kontrol a stanovíšť každej kontroly;
- premenné pre určenie aktuálneho času výpočtu od štartu aj od cieľa, doteraz nazbieraných bodov;
- výsledkové premenné obsahujúce hodnoty dosiahnutých bodov, času a zoznamy výsledkových prvkov;
- logické premenné určujúce, ktorý výpočet optimálnej cesty sa má použiť alebo pomáhaju pri tvorení grafu a nedovolia spustiť výpočet, kým nie je zadaný štart a čieľ v grafe;
- pomocné slovníky určujúce absolvovanie kontrol, súradníc vrcholov v bitmape, zoznamy a zásobníky čiastkových výsledkov a ostatné pomocné premenné;
- konštruktor a metódy na výpočet.

2.2.1 Metódy spracovania dát

Väčšina dát odovzdávame triedou *UserInterface*. Výnimkou je načítavanie dát zo súboru, ktoré po úvodnom dialógu v triede *UserInterface* pokračuje pomocou odkazu na vstupný súbor v triede *Solver*. Metóda *NacitajZoSuboru* načíta všetky informácie o vrcholoch, ich susedoch a vytvorí dátovú štruktúru *vrcholy*. Načítavanie sme zaobalili odchytom výnimiek, ktoré môžu nastať pri nekorektnom vstupnom súbore.

Medzi zadaním dát a spustením výpočtu prebehne v programe metóda *VyplnZoznamSpatnychSusedov*, ktorá prejde celý zoznam vrcholov a každému vrcholu

pripojí informáciu o hranách vstupujúcich do tohoto vrcholu spolu s ich ohodnotením. Po dobehnutí tejto metódy sa volá metóda *PrevedVypocet*, ktorá doplní potrebné premenné a spustí vybrané algoritmy hľadania optimálnej cesty.

Po ukončení algoritmov výpočtu z triedy *UserInterface* voláme metódu *NachystajVysledky*, ktorá prekontroluje a spočíta pozbierané body. Takisto nastaví dosiahnutý čas do premenných prístupných pre *UserInterface*, aby sme tieto údaje mohli zobraziť či uložiť do súboru pomocou *UserInterface*.

2.2.2 Metódy výpočtu podľa časového okienka

Výpočet podľa okienka začína metódov *VypocetPodlaCasu*, kde ako parametry odovzdávame štart, cieľ a dĺžku okienka. Na úvod cesty sa vloží prvý výsledok obsahujúci štart v čase 0 a zavolá sa metóda *FrontFirst* s rovnakými parametrami.

Primárny cyklus algoritmu (1.2.1) prebieha v metóde *FrontFirst*. Každý prechod cyklu obsahuje rekurzívne hľadanie optimálnej cesty cez každého suseda pomocou rekurzívnej metódy *FrontEnd*. Po prehľadaní ciest cez každého suseda sa vybraný sused nastaví ako nový výpočetný vrchol, posunie sa aktuálny čas. V prípade dostupnej kontroly sa pripočítajú na nej získané body a poznamená sa informácia o absolvovaní kontroly v slovníku použitia kontrol. Do zoznamu výsledkov sa pridá nový výsledok s aktuálnym výpočtovým vrcholom a časom.

Rekurzívna metóda *FrontEnd* zabezpečuje prehľadanie grafu. V parametroch obsahuje informácie o aktuálnom vrchole, čase, dĺžke presunu, nazbieraných bodoch na tejto ceste a zvyšnej veľkosti časového okienka. Pri odrazení od dna rekurzie porovná výsledky a určuje zlepšenie.

Na podobnom princípe funguje aj rekurzívna metóda *BackEnd*, ale prehľadáva graf od záveru s pomocou zoznamov spätných susedov a aktuálneho času výpočtu od konca. Používa rovnaké časové okienko ako *FrontEnd*.

Po ukončení cyklu v metóde *FrontFirst* preberá prácu metóda *Dual*, ktorá vo svojom tele obsahuje volania metód *FrontEnd* a *BackEnd*. Vrchol vypočítaný metódou *BackEnd* zaznamenáme do cesty len ak získaná čiastková cesta zozbiera body. V závere sa nastavia nové výpočtové vrcholy a metóda *Dual* sa rekurzívne zavolá znova s novými vrcholmi v parametroch. Jednou podmienkou na ukončenie rekurzie je pozbieranie všetkých bodov. Druhou podmienkou je pokles zvyšného času na hľadanie vrcholov pod hodnotu časového okienka. Po naplnení niektorej z podmienok preberá prácu metóda *DualWork* s parametrami aktuálneho konca cesty od štartu aj od cieľa.

Úlohou metódy *DualWork* je nájsť čo najkratšiu cestu medzi obdržanými vrcholmi v parametroch, keďže už máme vyzbierané všetky body, alebo zostá-

va málo času. Cestu hľadá pomocou prehľadávania do hĺbky s obmedzujúcou podmienkou navyše, aby cesta netrvala dlhšie ako je náš zvyšný čas. Algoritmus prehľadávania do hĺbky s danou podmienkou navyše sme naimplementovali v metóde *DFS*.

2.2.3 Metódy výpočtu podľa vybraných kontrol

Priebeh algoritmu (1.3.1) začíname už v metóde *VypocetPodlaBodov*, kde v úvode prebieha detekcia, priradenie do polí aj volanie metódy *VytvorKombinacie*, ktorá vytvorí spomínaný zoznam všetkých kombinácií. Pre každú kombináciu sa volá metóda *PocitajPodlaVrcholov*, a po jej skončení sa porovnávajú dosiahnuté výsledky.

PocitajPodlaVrcholov obsahuje priradenie štartu a cieľa do zoznamu aktuálnej kombinácie a iteráciu hľadania cesty medzi nasledujúcimi vrcholmi v kombinácii pomocou rekurzívnej metódy *NajdiCestu* a zároveň určuje maximálny počet zanorení.

NajdiCestu na úvod overí podmienky ukončenia rekurzie a možnosť zisku bodov na vrchole. Rekurziu ukončuje maximálna povolená hĺbka, nájdenie cieľového vrcholu alebo prekročenie času uzatvorenia cieľového vrcholu. Podobne ako *DFS*, rekurzívne hľadá cestu medzi susedmi, ale namiesto zvyšného času odovzdáva zostávajúci prípustný počet zanorení.

2.3 Trieda UserInterface

UserInterface slúži na základnú komunikáciu medzi programom a užívateľom. Zobrazuje príslušné informácie, okná a tlačidlá, kontroluje korektnosť niektorých vstupov a predáva získané informácie triede *Solver* za účelom výpočtu. Takisto preberá vypočítané výsledky od triedy *Solver* a zobrazuje ich užívateľovi.

2.3.1 Metódy zadávania dát

Tieto metódy môžeme rozdeliť na tri hlavné skupiny. V prvej skupine sa objavujú metódy zobrazujúce príslušné okienka, návestia so správnymi textami a tlačidlá, takisto ich vzájomnú previazanosť. Metódy *but_noveZad_Click*, *but_zmenaZad_Click* a *but_grafZad_Click* patria medzi základ zobrazovania v závislosti na výbere v hlavnom menu programu. Metóda *Hider* má na starosti zakrytie doteraz používaných okienok či návesti, používa sa na úvod každej zobrazovacej metódy alebo pri metóde *but_storno_Click*, aby príslušné metódy zobrazili správne úda-

je. Patrí sem aj metóda *ZobrazPridanieVrcholu*, ktorá má na starosti prechod od obecných dát k zadávaniu vrcholov.

Druhou skupinou sú metódy, ktoré okrem zobrazovania majú na starosti aj spracovanie a kontrolu korektnosti zadaných dát alebo len spracuvávajú dáta bez starostí o vzhľad programu. Sem patrí *but_pokr_Click*, ktorá spracuje obecné data nového zadania, odovzdá ich triede *Solver* a zavolá metódu *ZobrazPridanie-Vrcholu*. Metóda *but_vrchol_Click* spracuje údaje o čase otvorenia a zatvorenia vrcholu, jeho počtu bodov a takisto rozlišuje kontrolné body od štartu, cieľa a križovatiek. Po spracovaní zobrazí potrebné informácie pre zadanie susedov daného vrcholu. Odtiaľto je metódou *but_PridajSuseda_Click* možné pridanie suseda k aktuálne riešenému vrcholu, ale aj cez *But_kompletVrchol_Click* odovzdanie vrcholu triede *Solver* a opätovné povolanie metódy *ZobrazPridanieVrcholu*.

Všetky tieto metódy kontrolujú korektnosť údajov zadaných užívateľom, aby nedošlo k pádu programu pri nemožnosti spracovať vstup.

Tretia skupina sú metódy na prácu so súborom vstupných dát. Patrí sem *cb_otvoritSubor_CheckedChanged*, ktorá spustí dialóg na výber súboru a kontrluje jeho príponu. Metódou *but_otvoritSubor_Click* sa vybraný vyhovujúci súbor otvorí a načítajú sa z neho obecné dáta pre výpočet. V prípade, že načítanie zo súboru ostane zakliknuté aj naďalej, *but_pokr_Click* neposunie užívateľa k zadávaniu vrcholov a ich susedov, ale odovzdá odkaz na otvorený súbor triede *Solver*, ktorá si z neho načíta dáta.

2.3.2 Metódy opravy dát

V programe sme naimplementovali aj možnosť upraviť zadané dáta. Slúžia k tomu metódy volané po výbere zmeny zadania z hlavného menu, kde samotná metóda *but_zmenaZad_Click* okrem zobrazenia predvyplní aj aktuálne obecné hodnoty. Zmenu vykoná *but_oprav_Click*, ktorá kontroluje korektnosť, odovzdá hodnoty triede *Solver* a zavolá zobrazovaciu metódu *ZobrazOpravuVrcholu*, takisto metódu *NacitajVrchol*, ktorá predvyplní aktuálne hodnoty k vrcholu. Počas opravy vrcholov je kedykoľvek možné nezávisle spustiť výpočet. Pomocou *but_opravDalsiVrchol_Click* sa volá metóda *ZapisVrcholu* na zmenu hodnôt pri vrchole a prechádza sa k ďalšiemu vrcholu. Podobne *but_prejdiKSusedom_Click* opraví hodnoty pri vrchole a *ZobrazOpravuSuseda* zobrazí dialóg na opravu suseda takisto s predvyplnenými hodnotami pomocou *NacitajSuseda*.

2.3.3 Metódy grafického režimu

Grafický režim ponúka spôsob zadávania dát pre výpočet pomocou zvoleného obrázku a tomuto spôsobu odpovedajúce metódy. Nájdeme tu metódu *HiderGraphic*, ktorá po zavolaní skryje všetky používané okienka a návestia grafického režimu. Metóda *pb_Mapa_MouseClick* reaguje na udalosť kliknutia do bitmapy, pripraví novú bitmapu so zakresleným vrcholom a sprístupní potrebné okienka na zadanie údajov k vrcholu tak, aby *but_G_PridajVrchol_Click* mohla tieto informácie spracovať a predať vrchol triede *Solver*. Po ukončení pridávania do vrcholov *but_G_prejdiKSusedom_Click* zobrazí a predvyplní údaje o prvej dvojici vrcholov k zadaniu hrany pomocou metódy *NacitajGSusedov*. Táto metóda vypočíta pomocou Pytagorovej vety vzdialenosť všetkých zadaných vrcholov od zadaného vrcholu a zoradí ich podľa tejto vzdialenosti. Zobrazí vrchol a najbližsieho suseda v mape. Metóda *but_G_pridajSuseda_Click* pridá suseda k aktuálnemu vrcholu a zobrazí na mape ďalšieho suseda na pridanie. Pridávanie susedov prerušíme metódou *but_G_pridajSuseda_Click*, ktorá sa posunie v zozname vrcholov na ďalši vrchol, vypočíta opäť pomocou *NacitajGSusedov* vzdialenosti jeho susedov a zobrazí aktuálne informácie v mape. Na záver k týmto metódam radíme *but_G_spustiVypocet_Click*, ktorá spustí výpočet so zadanými dátami. Špeciálnou metódou je *but_G_UlozZadanie_Click*, pomocou ktorej sa ukladá do súboru rozpracované zadanie z grafického režimu. Súbor má podobný formát ako štandardný vstupný súbor pre program. Obsahuje naviac informácie o polohe vrcholov na mape. Po načítaní týchto dát je možné pridávanie ďalších vrcholov a nových susedov.

2.3.4 Metódy zobrazenia výsledkov

Metóda *ZobrazGVysledky* preberá od triedy *Solver* vypočítaný zoznam vrcholov najlepšej cesty, ktorú zakreslí do bitmapy a zobrazí v programe. Od grafického zobrazenia výsledkov sa k štandardnému zobrazeniu dostaneme použitím metódy *but_G_prejdiStandard_Click*, na rovnaké miesto nás privedie aj metóda *ZobrazVysledky*. Zobrazenie výsledkov na obrazovke v programe vykoná *but_zobrazVysledky_Click*, ktorá preberie výsledky a postupne vytvára informačný text. V programe sme naimplementovali aj uloženie výsledkov do súboru vykonané metódou *but_UlozDoSuboru_Click*, ktorá otvorí dialóg na výber súboru pre uloženie výsledkov. Po vyhovujúcom zadaní výstupného suboru sa automaticky do neho zapíšu všetky výpočtom získané informácie.

2.4 Súbor ResourcesSK.resx

Tento súbor združuje všetky informácie a návestia zobrazované triedou *UserInterface*. Umožňuje jednoduchú zmenu a aktualizáciu jednotlivých textov. Je predprípravou na užívateľskú viacjazyčnosť programu.

3. Výsledky

Pre porovnanie naimplementovaných algoritmov sme vytvorili testovacie dáta podľa reálnych pretekov aj náhodných grafov. Počet vrcholov sa pohyboval medzi 50 a 65, dĺžka trvania pretekov bola nastavená na 60 minút. Sledovali sme dobu výpočtu algoritmov, získané body a čas behu navrhnutý programom. Testy sme vykonali na počítači s dvojjadrovým procesorom Intel Core 2 Duo s frekvenciou 2,0 Ghz a L2 cache o veľkosti 2 MB.

3.1 Doby výpočtu algoritmov

Algoritmus využívajúci časové okienko je závislý na veľkosti zadaného okienka a priemernej hodnote hrán v grafe, od ktorých sa odvíja priemerný počet rekurzívnych zanorení pri jednom kroku algoritmu. Pri priemernom počte rekurzií 3 až 5 prebieha výpočet v ráde sekúnd, pri počte 6 či 7 sa presúvame do časov v rádoch desiatok sekúnd. Pri vyššom počte zanorení trvá výpočet rádovo minúty a viac. Dobu výpočtu takisto predlžujú oblasti v grafe s nízkymi hodnotami hrán, ktoré spôsobujú hlbokú rekurziu. Doba záverečného hľadania cesty medzi posledným výpočetným vrcholom a cieľom nezávisí na veľkosti okienka, iba na vzdialenosti vrcholov v grafe. V súlade s časovou zložitosťou algoritmu môže táto doba výrazne predčiť samotný výpočet ciest.

Pri algoritme podľa najhodnotnejších kontrol závisí doba výpočtu na ich počte x a takisto na počte stanovíšť. Zároveň s rastúcim počtom kontrol rastie aj počet výpočtov cesty medzi jednotlivými dvojicami vybranej kombinácie. Pri obvyklej hodnote troch stanovíšť každej kontroly je počet kombinácií 3^x. Výsledky s dátami obsahujúce 3 vybrané kontroly sme získali za niekoľko sekúnd, cestu s piatimi vybranými kontrolami sme vypočítali v priemere za minútu. Pri siedmich vybraných kontrolách výpočet trval už takmer 20 minút, vrátane obmedzujúcej podmienky na maximálne 8 zanorení rekurzie.

3.2 Výsledky algoritmov

Oba algoritmy nemajú hranicu aproximácie v najhoršom možnom prípade, výsledky závisia na vstupnom grafe a jeho vlastnostiach. V oboch prípadoch je možné, že algoritmus nevyzbiera žiadne body napriek tomu, že optimálna bodovaná cesta bude existovať.

Porovnávanie oboch algoritmov s optimálnym riešením je náročné pre zložitosť výpočtu daného optimálneho riešenia. Veľkú rolu hraje špecifický pohyb kontrol po rôznych stanovištiach a prispôsobenie sa algoritmov tomuto faktu. Viacero ekvivalentných bodov výrazne zvyšuje podiel vrcholov, ktoré majú v danom momente nulové ohodnotenie. Keďže žiadna kontrola nemôže byť na dvoch miestach súčasne, podiel bodovaných vrcholov otvorených v každom momente je nepriamo úmerný počtu rôznych stanovíšť. Zároveň počet stanovíšť určuje nároky na celkový počet vrcholov. To výrazne vplýva na exponenciálnu zložitosť prehľadávania všetkých ciest. Pre grafy vyhovujúce tomuto modelu, s prijateľnou dobou behu algoritmov, je výpočet optimálneho riešenia prakticky nemožný. Grafy s prijateľnou dobou výpočtu optimálneho riešenia obsahujú málo rôznych kontrol na relevantné posúdenie navrhnutých algoritmov. Redukciou na jedno stanovište každej kontroly, teda orientačný beh bez migrácie kontrol, sa stráca špecifickosť tohoto problému.

Algoritmus s časovým okienkom je vhodný pre grafy s nízkymi hodnotami hrán v porovnaní s celkovým časom pretekov a veľkosťou okienka. Prirovnaním sú dobre pripravení rýchli bežci, schopní sa dynamicky prispôsobovať. Pre algoritmus sú vhodné grafy s čo najvyšším podielom otvorených bodovaných kontrol. Výhodou je, ak je každá kontrola, nezávisle na stanovišti, otvorená čo najviac z celkovej doby pretekov. Nevhodné sú grafy s vysokou hodnotou hrán v porovnaní s okienkom. V týchto grafoch je pre malú hĺbku rekurzie prehľad o cestách hodne obmedzený a pohyb je často náhodný.

Počítanie s pomocou vybraných kontrol je vhodnejšie pre pomalších, rekreačných bežcov. Pre grafy s vysokou hodnotou hrán v porovnaní s dobou trvania. Kedže hĺbka rekurzie je pevne stanovená, pri malom podieli vybraných kontrol nie je možné pokrytie celej mapy, čo znižuje úspešnosť výsledkov. Z hľadiska výsledkov (nie doby trvania algoritmu) je vhodný pre grafy s vyšším podielom vybraných kontrol a najmä pre grafy, kde bežec nemá veľa času odbočovať z priamej cesty medzi dvojicou vybraných kontrol.

Pri výpočtoch v grafoch, kde existuje cesta zbierajúca všetky dostupné body za čas menší ako dve tretiny dĺžky trvania, ponúka lepšie výsledky časové okienko. Pri výrazne nižšom čase zaručene vyzbiera všetky kontroly, na rozdiel od algoritmu s vybranými kontrolami, v ktorom sme obmedzili počet vrcholov na cestách medzi jednotlivými vybranými kontrolami hĺbkou rekurzie. Okrem toho aj čas behu je vyšší pri vybraných kontrolách, kedže je zhora obmedzený iba dĺžkou trvania pretekov.

V ostatných prípadoch, ktoré sa zároveň viac približujú reálnemu priebehu

pretekov, závisia výsledky na zvolení hrán a veľkosti okienka. V dĺžke času behu sa vo väčšine prípadov algoritmy nelíšia, keďze časové okienko v tomto prípade buduje cestu aj od cieľa. Rozdiel sme pozorovali v počte získaných bodov, kde najlepšie výsledky podľa časového okienka boli aspoň o 20 % vyššie ako podľa vybraných kontrol.

3.3 Porovnanie s reálnymi pretekmi

Na úvod musíme zdôrazniť, že počet získaných bodov závisí hlavne na subjektívnom ohodnotení hrán, obzvlášť ak ich porovnávame s reálnymi výsledkami uskutočnených pretekov. Dáta z reálnych pretekov boli vytvorené s predpokladmi priemerného času absolvovania vzdialenosti 1 kilometer za 6 minút s prihliadaním na terén. Pripočítali alebo odpočítali sme 5 sekúnd za každých 5 výškových metrov, v závislosti na smere pohybu. K porovnaniu výsledkov s pretekmi sme vytvorili vstupné dáta z pretekov konaných v rokoch 2009, 2010 a 2011, všetky trvali 60 minút a časy dobehov boli vo väčsine tesne pred vypršaním tohoto limitu.

V roku 2009 by sa nami nasimulovaný bežec podľa algoritmu vybraných kontrol umiestnil s 25 bodmi z 37 možných na 48. mieste z 271 účastníkov, mužov aj žien [5]. Výsledky podľa časového okienka sú zaujímavejšie. S okienkom 5 minút dobehneme skôr ako víťaz, ale iba s 33 bodmi, čo stačí na 9. miesto. Okienko v hodnote 7 minút už pozbiera všetky kontroly a umiestni sa na 7. mieste, 8 minút dlhé okienko zabezpečí priečku o jedno miesto vyššiu.

Rok 2010 bol špecifický veľkou oblasťou s takmer nulovým prevýšením. Algoritmus vybraných kontrol by stačil na 18 bodov z 36 možných a 52. miesto z 278 účastníkov [5]. Tento rok je príkladom grafu, kde časové okienko narazilo na limit svojich možností. Nezávisle od nastavenia 5, 7 alebo 9 minút vyzbiera rovnaký počet bodov s drobnými časovými odlišnosťami a umiestni sa na 15. mieste spoločnej výsledkovej listiny s 26 bodmi. Tento graf je príkladom, kde neexistuje cesta zbierajúca všetky kontroly v časovom limite. Musíme dodať, že víťaz pretekov odbehol v rovnom teréne 1 kilometer za dobu 4 minúty a 30 sekúnd [5], naše dáta predpokladajú spomínaných 6 minút na kilometer.

Na záver sme pripravili porovnanie s rokom 2011. Algoritmus vybraných kontrol sa tradične pohybuje na úrovni 50. miesta, tentokrát 49. miesto z 269 účastníkov so ziskom 28 bodov z 41 možných [5]. Pri okienku veľkom 4 minúty získame 32 bodov a 12. miesto, na 5. miesto so ziskom 37 bodov nás dostane okienko veľké 6 minút.

Záver

Naimplementovali sme algoritmy časového okienka a vybraných kontrol spolu s užívateľským prostredím na zadávanie dát, ich úpravu a zobrazenie výsledkov. Porovnaním výsledkov sme zistili, že cesta podľa algoritmu vybraných kontrol sa neblíži k potenciálne najoptimálnejším cestám v grafe. Hlavným dôvodom je obmedzenie rekurzie medzi jednotlivými krokmi algoritmu, aby sme zachovali primeranú dobu výpočtu. Cesta podľa algoritmu časového okienka získa viac bodov. Jej výsledky sa v grafových modeloch reprezentujúcich reálne preteky, kde sme zadali odhady presunov podľa reálnych schopností bežcov, blížia najlepším výsledkom dosiahnutým v pretekoch, prípadne ich vyrovnávajú. Doba výpočtu u oboch algoritmov závisí od počtu vrcholov a zadaných parametrov, najmä časového okienka a počtu vybraných kontrol. Výsledky sú obvykle dostupné v priebehu desiatok sekúnd až rádovo v minútach.

Zoznam použitej literatúry

[1] BANSAL, Nikhil, BLUM, Avrim, CHAWLA, Shuchi, MEYERSON, Adam. *Approximation algorithms for deadline-TSP and vehicle routing with time-windows.* V *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, str. 166 – 174. New York: ACM, 2004.

[2] CHEKURI, Chandra, KORULA, Nitish, PÁL, Martin. *Improved Algorithms for Orienteering and Related Problems.* V *Transactions on Algorithms Volume 8 Issue 3*, čl. 23. New York: ACM, 2012.

[3] CHEKURI, Chandra, PÁL, Martin. *A Recursive Greedy Algorithm for Walks in Directed Graphs.* V *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, str. 245 - 253. Washington: IEEE Computer Society, 2005.

[4] KANTOR, Marisa G., ROSENWEIN, Moshe B. *The Orienteering Problem with Time Windows.* V *Journal of the Operational Research Society Vol. 43, No. 6*, str. 629 – 635. Great Britain: Palgrave Macmillan Journals, 1992.

[5] SAJAL, Martin. *SMIK - OB S MIgrujícími Kontrolami.* Dostupné na: http://ob.vse.cz/smik/

[6] SHMOYS, David B., WILLIAMSON, David P. *The Design of Approximation Algorithms.* 1. vydanie. Cambridge: Cambridge University Press, 2011. ISBN 978-0-521-19527-0

[7] TASGETIREN, M. Fatih. *A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem.* V *Journal of Economic and Social Research 4 (2)*, str. 1 – 26. r. 2005. Dostupné na: http://www.fatih.edu.tr/ jesr/tasgetiren.pdf

[8] *Part 2: The Queue, Stack, and Hashtable.* Dostupné na: http://msdn.microsoft.com/en-us/library /ms379571%28v=vs.80%29.aspx

Príloha – Užívateľská dokumentácia

SMIK solver hľadá optimálnu cestu orientovaného grafu s ohodnotenými vrcholmi. Je určený pre operačné systémy Windows. K dispozícii je inštalačný balíček, projekt z Visual Studia a vzorový súbor s testovacími dátami.

Inštalácia programu

V priečinku *Install* nájdeme súbor *setup.exe*. Po jeho spustení potvrdíme nainštalovanie a program sa po dokončení inštalácie spustí. Opätovné spustenie je možné medzi programami v ponuke Štart operačného systému.

Odinštalovanie programu prebieha v ovládacích paneloch cez záložku *Pridať alebo odobrať programy*.

Užívateľský návod

Po spustení programu sa zobrazí základné okno s možnosťami na ľavej strane (obrázok 1).

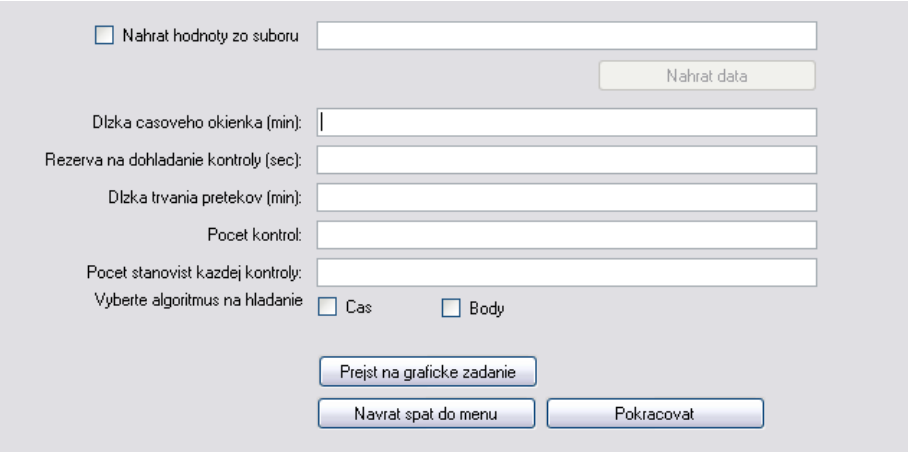
Obr. 1: Úvodné okno programu

Zadanie vstupných dát

Program ponúka 3 základné možnosti zadania vstupných dát: individuálne pre každé stanovište zvlášť, načítanie dát z kompatibilného a korektného súboru alebo zadanie pomocou grafickej mapy. Po zadaní programom požadovaných dát je možné prejsť k výpočtu.

Zadanie základných informácií o behu

Po kliknutí na nové zadanie sa zobrazí okno na vyplnenie základných informácií o behu (obrázok 2).



Obr. 2: Zadávanie obecných hodnôt

Môžeme načítať dáta zo súboru alebo ich vyplniť osobne. Dĺžka časového okienka je celočíselná hodnota vyjadrujúca trvanie v minútach. Rezerva na dohľadanie kontroly počíta s možnou nepresnosťou pri behu. V prípade, že sa stanovište zatvára v období medzi dobehnutím na miesto a časom s rezervou, tak užívateľ na výstupe dostane upozornenie. Rezerva sa zadáva celočíselne v sekundách. Trvanie pretekov je časový limit na dobehnutie do cieľa bez penalizácie, zadáva sa celočíselne v minútach. Počet stanovíšť určuje počet rôznych miest každej kontroly. Je potrebné zadať aspoň jednu metódu na výpočet, pre porovanie je vhodné vypočítať cestu oboma spôsobmi.

Po zadaní týchto základných informácií môžeme prejsť ku grafickému zadaniu vrcholov alebo pokračovať v zadávaní bez mapy.

Načítanie dát zo súboru

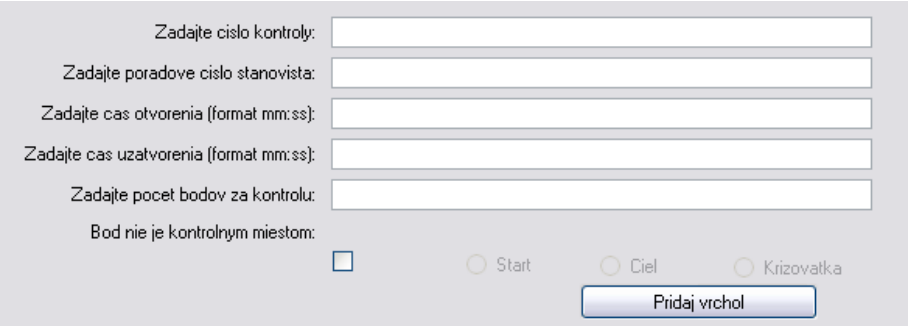
Načítanie zo súboru umožnuje najrýchlejšie a najjednoduchšie zadanie dát pre výpočet. Zakliknutím okienka pri obecnom zadávaní otvoríme dialóg na výber súboru. Je nutné, aby súbor končil príponou *.smik* alebo *.smkg* pre pokračovanie v grafickom režime. Následne je ponúknutá možnosť načítať obecné dáta cez tlačidlo *Nahrat data*. Po nahraní môžeme meniť obecné data, s ktorými bude program počítať nezávisle od súboru. Kliknutie na pokračovanie načíta automaticky dáta o vrcholoch a ich susedoch. Ak chceme zo súboru použit iba obecné hodnoty a následne vrcholy zadať osobne, odklikneme načítanie dát zo súboru.

V tom prípade sa súbor nepoužije a budeme presmerovaný na štandardné zadávanie vrcholov.

Dáta zo súboru s grafickým rozšírením možno používať aj vo výpočetnom móde bez grafického režimu. V prípade presunu na grafický mód je po výbere mapy užúvateľovi ponúknutá možnosť načítania dát zo súboru.

Štandardné zadávanie vrcholov

Ďalším krokom je zadanie vrcholov (obrázok 3).



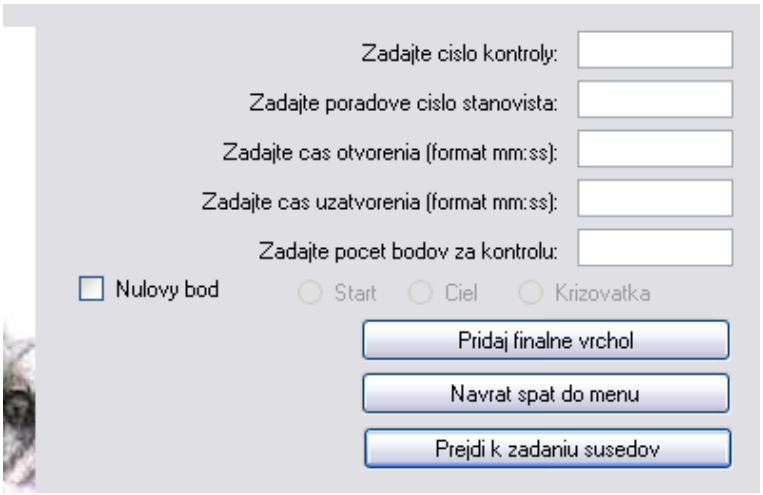
Obr. 3: Zadávanie vrcholu

Vrchol je každé miesto v mape, ktoré sa rozhodneme zadať pre výpočet. Okrem konkrétnych kombinácií kontrol a stanovišťa sa jedná aj o nehodnotené vrcholy - štart, cieľ a významné body na mape, križovatky. Nehodnotené vrcholy majú číslo kontroly 0. Pre zadanie takého vrcholu zaklikneme možnosť, že bod nie je kontrolným miestom a vyberieme jeho kategóriu. V prípade, že sa jedná o štart nie je potrebné zadávať nič iné. V prípade konca či križovatky zadáme iba poradové číslo stanovišťa. Pri zadávaní vrcholu štandardného kontrolného miesta vyplníme žiadané údaje.

Každý vrchol má svojich susedov - iné vrcholy v jeho okolí. Po pridaní vrcholu sme presmerovaní na zadávanie susedov. Môžeme zadať kontrolné aj nekontrolné body a odhadovaný čas presunu na dané miesto. Kliknutím na pridanie suseda vytvoríme jednosmerné spojenie medzi danými vrcholmi. Po kliknutí na finálne pridanie vrcholu budeme presmerovaní na zadanie nového vrcholu.

Grafické zadávanie vrcholov

Zo zadania obecných hodnôt môžeme prejsť na grafické zadanie. Zadáme obrázok mapy vo formáte *.bmp*, *.jpg*, *.gif* alebo *.png*. Maximálna veľkosť obrázku je 800x600, väčsie obrázky program oreže podľa týchto parametrov. Po jeho zadaní môžeme pridávať vrcholy (obrázok 4).



Obr. 4: Grafické zadanie

Po kliknutí do mapy môžeme vyplniť údaje o vrchole. Vrchol sa pridá stále v mieste posledného kliknutia, jeho zobrazenie v mape prebehne až po pridaní vrcholu tlačidlom. Na rozdiel od štandardného zadávania vrcholov sa nestrieda zadávanie vrcholu s jeho susedmi.

Zadávanie susedov prebieha až po naklikaní všetkých vrcholov do mapy. Program postupne ponúka a zobrazuje na pridanie susedov od najbližšie umiestneného vrcholu po najvzdialenejší. V prípade, že nechceme zadať daný vrchol ako suseda, políčko so vzdialenosťou necháme prázdne a kliknutím na tlačítko program ponúkne ďalšieho suseda. Ak sme zadali všetkých susedov o ktorých bol záujem, alebo program vyčerpal všetky možné dvojice s daným vrcholom, môžeme prejsť na ďalší vrchol.

V novej verzii programu je umožnené ukladanie naklikaných dát spolu s informáciami o pozícii vrcholov v mape do súboru. Obnovenie týchto dát sa vykonáva cez nové zadanie – súbor – prechod na grafický režim, kde je po nahratí mapy umožnené načítanie dát.

Zmena zadaných dát

Pri štandardnom zadávaní dát môžeme po zadaní dát pred spustením výpočtu jednotlivé dáta upraviť. Takisto môžeme dáta upraviť po výpočte, ale s upravenými dátami už nie je možný návrat do zobrazenia výsledkov v grafickom režime.

Na daný úkon slúži zmena zadania z hlavnej ponuky programu. Môžeme zmeniť obecné hodnoty pre výpočet, ale rovnako aj hodnoty jednotlivých vrcholov. Nie je možné pridávať ďalších susedov vrcholu, ale zadaním času presunu väčšieho ako dĺžka trvania pretekov môžeme daného suseda prakticky vynechať z výpočtu.

Kedykoľvek počas úpravy dát môžeme spustiť nový výpočet.

Výpočet najlepšej cesty

Počas výpočtu sú všetky možnosti neaktívne. Po ukončení výpočtu program ponúkne najlepšie nájdené výsledky podľa každej zo zvolených metód. V prípade neúmerne veľkého časového okienka a rýchlych presunov medzi jednotlivými vrcholmi môže výpočet trvať rádovo desiatky minút a viac.

Zobrazenie výsledkov

Po dokončení výpočtu zobrazíme výsledky v závislosti na spôsobe zadávania dát.

Grafické zobrazenie

Grafické výsledky sú zobrazené v mape jednotlivými čiarami medzi vrcholmi. V prípade, že užívateľ zadal obe metódy výpočtu sú ako prvé výsledky zobrazené podľa časového okienka. Kliknutím na tlačítko je možné prejsť na zobrazenie druhej možnosti alebo na ponuku štandardného zobrazenia.

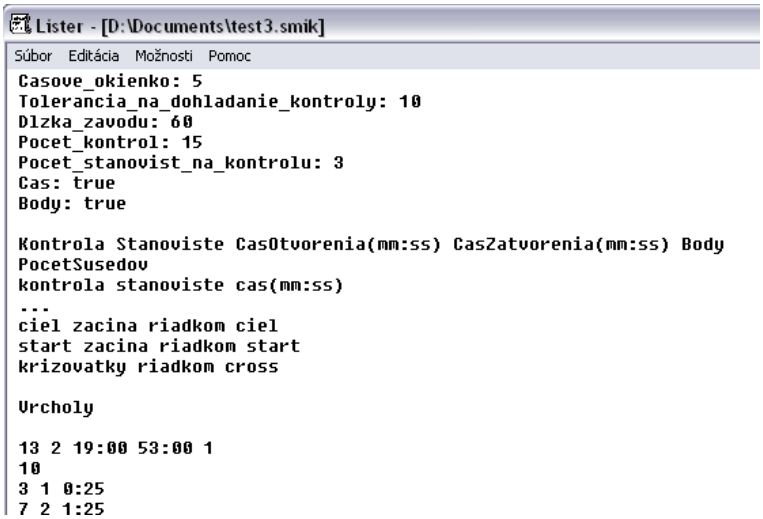
Štandardné zobrazenie

Program ponúka priame zobrazenie výsledkov rovnako ako aj ich uloženie do súboru. Vo výsledkoch je uvedený čas, za ktorý bežec dobehol, nazbierané body a zoznam jednotlivých vrcholov s očakávanými časmi prechodu. Zoznam sa zobrazuje postupne po desiatich vrcholoch, kliknutie zobrazí ďalšiu desiatku alebo prejde na zobrazenie výsledkov druhej metódy.

Súbory vstupných dát

Pre jednoduchšiu prácu s programom slúžia textové súbory s nachystanými vstupnými dátami končiace príponou *.smik* a *.smkg*. *SMIK solver* nekontroluje korektnosť zadaného súboru, iba príponu a predpokladá korektnosť dat vo vyhovujúcich súboroch. Po načítaní program zobrazí hlášku, buď o úspechu alebo upozorní na problém. Počítanie s nekorektnými dátami môže spôsobiť pád programu.

Prvých sedem riadkov súboru sú obecné informácie v poradí dĺžka okienka, tolerancia, dĺžka pretekov, počet kontrol, počet stanovíšť, žiadosť o výpočet podla času a podľa bodov. Po prvej medzere by mala nasledovať žiadaná hodnota, teda číslo alebo slová *true* či *false*. Až do riadku obsahujúceho samostatné slovo *Vrcholy* môže súbor obsahovať ľubovoľné informácie. Po riadku so slovom *Vrcholy*



Obr. 5: Ukážka vstupného súboru

nasleduje medzera a až do konca súboru medzerou oddelené jednotlivé vrcholy (na obrázku 5).

Zadanie vrcholu pozostáva z riadku obsahujúceho číslo kontroly, stanovišťa, času otvorenia a zatvorenia vo formáte *mm:ss* a počtu bodov za daný vrchol. V prípade *.smkg* súborov tento riadok ešte obsahuje súradnice x a y vyjadrujúce pozíciu na mape v pixeloch. Jednotlivé údaje sú oddelené medzerou. Ak sa nejedná o kontrolný bod, riadok začína slovom *start* pre štart a číslom 0, *ciel* pre cieľový vrchol a číslo jeho poradia, ostatné body slovom *cross* a číslom ich poradia. Nasleduje riadok obsahujúci počet susedov a následne daný počet riadkov so susedmi vo formáte číslo kontroly, stanovišťa suseda a času presunu vo formáte *mm:ss*. Po poslednom susedovi je prázdny riadok a nasleduje ďalší vrchol.

Tipy

V niektorých pretekoch majú rôzne kontroly rovnakú pozíciu v mape. Odporúčame jeden z týchto združených vrcholov označiť ako vstupný a do neho viesť všetky hrany od susedov. Vstupnému vrcholu nastavíme práve jedného suseda – vrchol na rovnakom mieste vo vzdialenosti 0. Ak sú na mieste viac ako dva vrcholy, postupne medzi nimi nastavíme cestu až po posledný vrchol, z ktorého vedú cesty na ostatných susedov v okolí. Týmto spôsobom zaručíme, že program vezme v úvahu všetky prítomné kontroly a zároveň zabránime nekonečnej rekurzii pri ceste medzi vrcholmi so vzdialenosťou 0.