EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 13006/B/2025/36146475535437572

NÁVRH INTERAKTÍVNEJ APLIKÁCIE SHINY V JAZYKU R NA RIEŠENIE A VIZUALIZÁCIU VYBRANÝCH ÚLOH

Bakalárska práca

2025 **Martina Lábska**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

NÁVRH INTERAKTÍVNEJ APLIKÁCIE SHINY V JAZYKU R NA RIEŠENIE A VIZUALIZÁCIU VYBRANÝCH ÚLOH

Bakalárska práca

Študijný program: Hospodárska informatika

Študijný odbor: Ekonómia a manažment

Školiace stredisko: Katedra matematiky a aktuárstva

Vedúci záverečnej práce: doc. Ing. Michal Páleš, PhD.

Bratislava 2025 **Martina Lábska**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Martina Lábska
Študijný program:	hospodárska informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	8. - ekonómia a manažment
Typ záverečnej práce:	Bakalárska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Návrh interaktívnej aplikácie Shiny v jazyku R na riešenie a vizualizáciu vybraných úloh

Anotácia: Bakalárska práca sa bude zaoberať návrhom aplikácie Shiny v jazyku R, pričom jej funkcionalita bude využitá v oblasti (zodpovedajúcej problematike štúdia I. stupňa), ktorú si autor vyberie. Autor opíše postup vytvárania navrhnutej aplikácie a tiež výhody a nevýhody jej použitia. Od študenta sa požaduje cieľavedomosť pri štúdiu informácií, jednak z oblasti Shiny v jazyku R a jednak z oblasti, ktorú bude pomocou Shiny aplikácie spracovávať.

Vedúci:	doc. Ing. Michal Páleš, PhD.
Katedra:	KMA FHI - Katedra matematiky a aktuárstva
Vedúci katedry:	doc. Ing. Michal Páleš, PhD.
Dátum zadania:	01.03.2024

Dátum schválenia: 11.03.2024

doc. Ing. Martin Mišút, CSc.
osoba zodpovedná za realizáciu študijného programu

ABSTRAKT

LÁBSKA, Martina: Návrh interaktívnej aplikácie Shiny v jazyku R na riešenie a vizualizáciu vybraných úloh – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra matematiky a aktuárstva. – Vedúci záverečnej práce: doc. Ing. Michal Páleš, PhD. – Bratislava: FHI, 2025, 82s.

Cieľom záverečnej práce je vytvoriť interaktívnu aplikáciu Shiny v jazyku R na riešenie a vizualizáciu vybraných úloh. Ako oblasť spracovávaná aplikáciou bola zvolená problematika Management Science. Práca je rozdelená do štyroch častí. Obsahuje 10 tabuliek, 39 obrázkov a 1 prílohu. Prvá časť sa venuje využitiu aplikácií Shiny a teoretickému vymedzeniu Management Science a optimalizačných úloh. V druhej časti je spracovaný hlavný cieľ práce a čiastkové ciele na jeho dosiahnutie. Predposledná kapitola je zameraná na teoretické a aplikačné riešenie vybraných úloh – dopravná úloha, výrobný problém, hľadanie najkratšej a okružnej cesty. V záverečnej časti skúmame funkčnosť aplikácie na konkrétnych príkladoch, pričom hodnotíme jej silné a slabé stránky. Výsledkom práce je prehľadná interaktívna aplikácia s vysokým potenciálom učebnej pomôcky.

Kľúčové slová: R Shiny, Shiny aplikácia, Management Science, optimalizačné úlohy, dopravná úloha, výrobný problém, hľadanie najkratšej cesty, úloha obchodného cestujúceho

ABSTRACT

LÁBSKA, Martina: Design of an interactive Shiny application in R for solving and visualizing selected problems – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Mathematics and Actuarial Science. – Thesis supervisor: doc. Ing. Michal Páleš, PhD. – Bratislava: FHI, 2025, 82p.

The aim of this thesis was to develop an interactive Shiny application in R for solving and visualizing selected problems. The field of implementation selected for the application was Management Science. The thesis is divided into four sections and includes 10 tables, 39 pictures, and 1 attachment. The first section focuses on the use of Shiny in R and the theoretical background of Management Science and optimization problems. The second section defines the main objective of the thesis and outlines the partial goals needed to achieve it. The penultimate chapter is dedicated to the theoretical and practical solutions of selected problems – transportation problem, production problem, shortest path finding, and the traveling salesman problem. In the final section, we analyse the functionality of the application using practical examples and evaluate its strengths and weaknesses. The outcome of the thesis is a clear and interactive application with high potential as an educational tool.

Key words: R Shiny, Shiny application, Management Science, optimization problems, transportation problem, production problem, shortest path finding, traveling salesman problem

OBSAH

ÚVOD **9**
1 SÚČASTNÝ STAV RIEŠENEJ PROBLEMATIKY DOMA A V ZAHRANIČÍ **10**
1.1 Aplikácie Shiny 10
1.1.1 Využitie Shiny aplikácií v praxi 11
1.1.2 Reaktívne programovanie 13
1.1.3 Rozširujúce knižnice pre Shiny 14
1.1.4 Možnosti zdieľania Shiny aplikácií 15
1.1.5 Shiny pre R vs. pre Python 17
1.2 Výhody a nevýhody R Shiny v porovnaní s inými nástrojmi 18
1.2.1 Nástroje Business Intelligence 18
1.2.2 Dash v jazyku Python vs. Shiny v R 20
1.3 Možnosti využitia Shiny v úlohách Management Science 21
1.3.1 Modelovanie podnikových procesov 23
1.3.2 Optimalizačné úlohy 25
1.3.3 Softvérové možnosti riešenia optimalizačných úloh 26
2 CIEĽ PRÁCE **32**
3 METODIKA PRÁCE A METÓDY SKÚMANIA **33**
3.1 Všeobecný proces tvorby Shiny aplikácie 33
3.1.1 UI (user interface) 34
3.1.2 Server 37
3.1.3 Reaktívne prvky 38
3.1.4 Shinydashboard 39
3.2 Charakteristika úloh Management Science 41
3.2.1 Výrobný problém 42
3.2.2 Dopravná úloha 44
3.2.3 Hľadanie najkratšej cesty 45
3.2.4 Hľadanie najkratšej okružnej cesty 48
3.3 Tvorba aplikácie Shiny 49
3.3.1 Výrobný problém 50

3.3.2 Dopravná úloha ... 52
3.3.3 Nájdenie najkratšej a najkratšej okružnej cesty ... 54
4 VÝSLEDKY PRÁCE ... 55
4.1 Overenie funkcionality vytvorenej aplikácie ... 55
4.1.1 Aplikácia: Dopravná úloha ... 55
4.1.2 Aplikácia: Výrobný problém ... 59
4.1.3 Aplikácia: Hľadanie najkratšej a najkratšej okružnej cesty ... 61
4.2 Výhody a nevýhody vytvorenej aplikácie ... 62
4.2.1 Výhody ... 63
4.2.2 Nevýhody ... 63
ZÁVER ... 64
ZOZNAM POUŽITEJ LITERATÚRY ... 65
PRÍLOHY ... 67

ZOZNAM TABULIEK A OBRÁZKOV

Tab. 1.2.1 Porovnanie R Shiny s nástrojmi Business Intelligence 18
Tab. 1.3.1 Klasifikácia metód na riešenie optimalizačných rozhodovacích úloh 23
Tab. 3.1.1 Prehľad vstupných elementov v Shiny 35
Tab. 3.1.2 Prehľad výstupných elementov v Shiny 36
Tab. 3.1.3 Typy boxov v shinydashboard 40
Tab. 3.2.1 Začiatočná simplexová tabuľka výrobného problému 43
Tab. 3.2.2 Optimálna tabuľka výrobného problému 43
Tab. 3.2.3 Najkratšie vzdialenosti z u_1 Dijkstrov algoritmus 47
Tab. 3.2.4 Tabuľka vzdialeností uzlov príklad algoritmu najbližšieho suseda 49
Tab. 4.1.1 Matica nákladov na prepravu jednej jednotky tovaru 56

Obr. 1.1.1 COVID-19 tracker aplikácia 11
Obr. 1.1.2 Watch Dashboard aplikácia 12
Obr. 1.1.3 Shiny MRI aplikácia 12
Obr. 1.1.4 shinyCircos aplikácia 13
Obr. 1.1.5 Porovnanie tradičného zdieľania Shiny a zdieľania pomocou Shinylive 16
Obr. 1.3.1 Schéma postupu modelovania. 25
Obr. 1.3.2 Príprava údajov pre riešenie ÚLP pomocou doplnku Riešiteľ 27
Obr. 1.3.3 Dialógové okno Parametre doplnku Riešiteľ 28
Obr. 1.3.4 Dialógové okno Výsledky doplnku Riešiteľ a riešenie ÚLP 29
Obr. 1.3.5 Kód riešenia ÚLP v jazyku R 30
Obr. 1.3.6 Výstup z konzoly R 30
Obr. 3.1.1 Kód jednoduchej Shiny aplikácie 33
Obr. 3.1.2 Informácie o pripojení aplikácie v konzole 33
Obr. 3.1.3 Tlačidlo pre spustenie aplikácie 34
Obr. 3.1.4 Ukážka aplikácie s navigačným panelom v hornej časti 34
Obr. 3.1.5 Ukážka vstupných elementov v Shiny 36
Obr. 3.1.6 Ukážka výstupných elementov v Shiny 37
Obr. 3.1.7 Kód základnej štruktúry dashboardu 39
Obr. 3.1.8 Vizualizácia boxov v shinydashboard 40

Obr. 3.1.9 Statusy boxov v shinydashboard, ... 41
Obr. 3.2.1 Orientovaný ohodnotený graf pre príklad Dijkstrovho algoritmu 48
Obr. 3.3.1 Riešenie pomocou lpSolve 51
Obr. 3.3.2 Riešenie dopravnej úlohy pomocou lp.transport() 53
Obr. 3.3.1 Úvodná stránka aplikácie 55
Obr. 4.1.1 Zadávanie vstupných údajov dopravná úloha 56
Obr. 4.1.2 Zadávanie vstupných údajov a nákladovej matice dopravná úloha 57
Obr. 4.1.3 Výber optimalizácie dopravná úloha 57
Obr. 4.1.4 Vizualizácia riešenia dopravnej úlohy 58
Obr. 4.1.5 Vizualizácia riešenia dopravnej úlohy - tepelná mapa 58
Obr. 4.1.6 Tabuľka a interpretácia riešenia dopravnej úlohy 58
Obr. 4.1.7 Kontrola riešenia dopravnej úlohy pomocou kódu v R 59
Obr. 4.1.8 Výsledok riešenia dopravnej úlohy v R konzole 59
Obr. 4.1.9 Zadávanie vstupných údajov pre výrobný problém 59
Obr. 4.1.10 Tabuľka, interpretácia, a graf riešenia výrobného problému 60
Obr. 4.1.11 Zadávanie spotreby a optimalizačného cieľa výrobný problém 60
Obr. 4.1.12 Vstupná tabuľka vzdialeností uzlov 61
Obr. 4.1.13 Kód vstupných údajov pre kontrolu v R 61
Obr. 4.1.14 Vygenerovaný graf a riešenie úloh najkratšej cesty a okružnej cesty 62
Obr. 4.1.15 Konzolový výstup výpočtu najkratšej cesty v R 62

ÚVOD

V súčasnosti zohráva tvorba interaktívnych webových aplikácií významnú úlohu pri vizualizácii dát a spracovaní výpočtov. Balík Shiny v jazyku R predstavuje efektívny nástroj na vývoj takýchto aplikácií aj bez potreby znalostí webových technológií, ako sú HTML či JavaScript. Vďaka svojej prehľadnej štruktúre a prepojeniu s výpočtovými možnosťami jazyka R umožňuje vytváranie komplexných a dynamických riešení, ktoré reagujú na vstupy používateľa v reálnom čase.

Ako oblasť implementácie aplikácie bola zvolená problematika Management Science, ktorá sa zameriava na využitie kvantitatívnych metód pri riešení komplexných manažérskych problémov. Ide o interdisciplinárny prístup kombinujúci matematické modely s efektívnym plánovaním zdrojov, riadením výroby, logistikou a optimalizáciou dopravy. Táto oblasť bola zvolená ako vhodná z hľadiska aplikačného potenciálu, so zámerom priblížiť jej podstatu aj budúcim študentom, ktorí sa ňou budú zaoberať.

Úvodná časť práce sa venuje možnostiam využitia Shiny aplikácií, ich výhodám a porovnaniu s alternatívnymi nástrojmi, ako sú Python Dash alebo nástroje Business Intelligence. Rozoberá sa tiež prehľad dostupných softvérových riešení pre optimalizačné úlohy, ako napríklad Excel či samotný jazyk R. Následne je definovaný hlavný cieľ bakalárskej práce spolu s jednotlivými čiastkovými cieľmi nevyhnutnými na jeho dosiahnutie.

V predposlednej časti sa podrobne analyzuje štruktúra a fungovanie Shiny aplikácií – opisuje sa používateľské rozhranie, serverová logika, typy vstupov a výstupov, ako aj využitie reaktívnych prvkov. Zároveň sú predstavené tradičné metódy riešenia vybraných optimalizačných úloh, medzi ktoré patria dopravná úloha, výrobný problém, hľadanie najkratšej cesty a hľadanie najkratšej okružnej cesty. Vysvetlené sú princípy simplexovej metódy, Dijkstrovho algoritmu a algoritmu najbližšieho suseda, ktoré sú následne doplnené o aplikačné spracovanie vrátane popisu vstupných údajov a výpočtového postupu.

Záverečná časť práce sa venuje testovaniu aplikácie na konkrétnych príkladoch a hodnoteniu jej funkčnosti, výhod a obmedzení. Výsledkom práce je prehľadná interaktívna aplikácia s potenciálom učebnej pomôcky, ktorá môže prispieť k lepšiemu porozumeniu optimalizačných metód.

1 SÚČASTNÝ STAV RIEŠENEJ PROBLEMATIKY DOMA A V ZAHRANIČÍ

1.1 Aplikácie Shiny

Shiny je interaktívny framework pre tvorbu webových aplikácií, ktorý bol vyvinutý spoločnosťou RStudio (dnes Posit). Jeho hlavný autor Joe Cheng ho predstavil pre verejnosť 31. Júla 2012 na konferencii JSM ako nástroj na jednoduché vytváranie aplikácií v jazyku R bez potreby znalostí HTML, JavaScript, alebo CSS.[1]

Názov *Shiny* bol inšpirovaný populárnym sci-fi seriálom *Firefly*, kde je slovo „shiny" spojené s niečím pozitívnym alebo dobrým.[2] Tento názov sa stal synonymom pre nástroj, ktorý zásadne zmenil prístup k dátovej analýze a vizualizácii, poskytujúc používateľom možnosť vytvárať a zdieľať „shiny" – profesionálne aplikácie bez nutnosti rozsiahlych technických vedomostí.

Hlavným impulzom pre vytvorenie Shiny bola snaha prekonať obmedzenia, ktorým čelila dátová veda, so zameraním na prezentáciu výsledkov. Analýza dát v samotnom R prebiehala prevažne staticky, napríklad pomocou balíkov `ggplot2` alebo `knitr`, často v podobe PDF alebo HTML správ. Tieto metódy síce efektívne zachytávali zistenia analytikov, no chýbala im interaktivita, ktorú poskytujú moderné webové aplikácie. Shiny bola vyvíjaná s myšlienkou poskytnúť interaktivitu a UI dizajn pre používateľov jazyka R, a aby ich práca od spracovania dát až po ich vizualizáciu mohla byť zastrešená iba jedným kódom v rámci prostredia R.

Jazyk R ako jeden z najpopulárnejších jazykov v oblasti štatistiky dátovej analýzy či bioinformatiky sa osvedčil ako efektívny a neustále napredujúci nástroj, avšak pred príchodom balíka Shiny sa iba zriedka používal na tvorbu webových aplikácií. Dnes už je nespočetne veľa projektov a aplikácií postavených na Shiny, keďže sa preukázal jej potenciál nie len pri spracovávaní údajov, ale aj vytváraní rozsiahlych databáz a serverov. V roku 2020 bola vytvorená

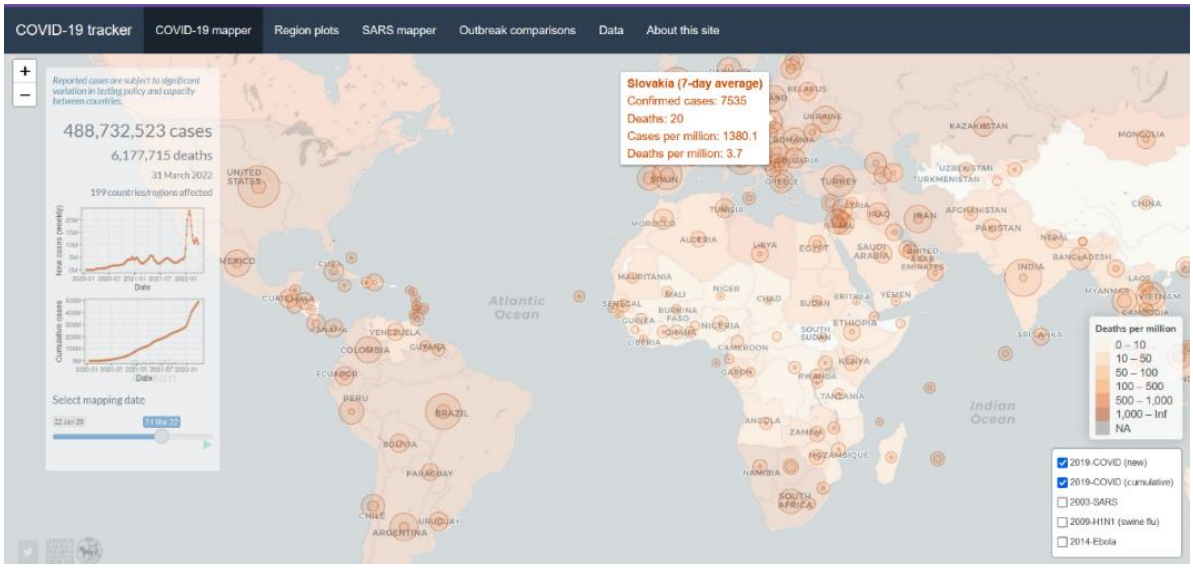
[1] KASPRZAK, Peter et al., 2020. Six Years of Shiny in Research - Collaborative Development of Web Tools in R. *The R Journal.* 12(2), s. 25. ISSN 2073-4859
[2] CHENG, Joe. *The Past and Future of Shiny* [video]. YouTube, 2022, min. 49.05 – 49.50 [cit. 19. 01. 2025]. Dostupné z: https://www.youtube.com/watch?v=HpqLXB_TnpI

interaktívna webová aplikácia Shiny na vizualizáciu dát o pandémii COVID-19 v Japonsku, ktorá zaznamenala viac ako 13 miliónov návštev.[3]

1.1.1 Využitie Shiny aplikácií v praxi

Rastúci počet úspešných implementácií potvrdzuje, že Shiny aplikácie dokážu transformovať výsledky analýz do podoby zrozumiteľnej aj pre širšie publikum, čím prispievajú k aktívnejšiemu zapojeniu používateľov do interpretácie dát. Vybrané príklady zobrazujú rozmanité spôsoby využitia balíka Shiny v praxi, pričom poukazujú na jeho flexibilitu a prispôsobivosť v rôznych odborných oblastiach:

- **COVID-19 Tracker** – jedna z najznámejších Shiny aplikácií, ktorá vizualizovala priebeh pandémie COVID-19 na základe údajov od Johns Hopkins University. Interaktívne grafy umožňovali sledovanie trendov, porovnávanie s predchádzajúcimi epidémiami a hodnotenie efektivity opatrení.

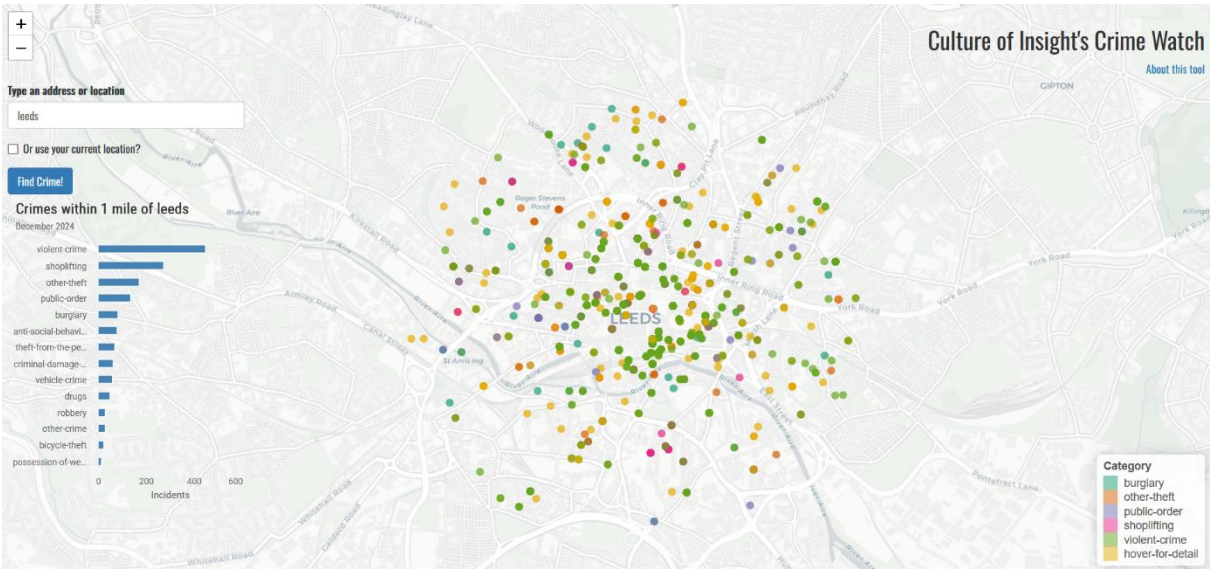


Obr. 1.1.1 COVID-19 tracker aplikácia
Zdroj: https://vac-lshtm.shinyapps.io/ncov_tracker/

- **Crime Watch Dashboard** – nástroj určený na analýzu kriminality v Spojenom kráľovstve. Používatelia môžu pomocou GPS údajov zobrazovať údaje o trestnej

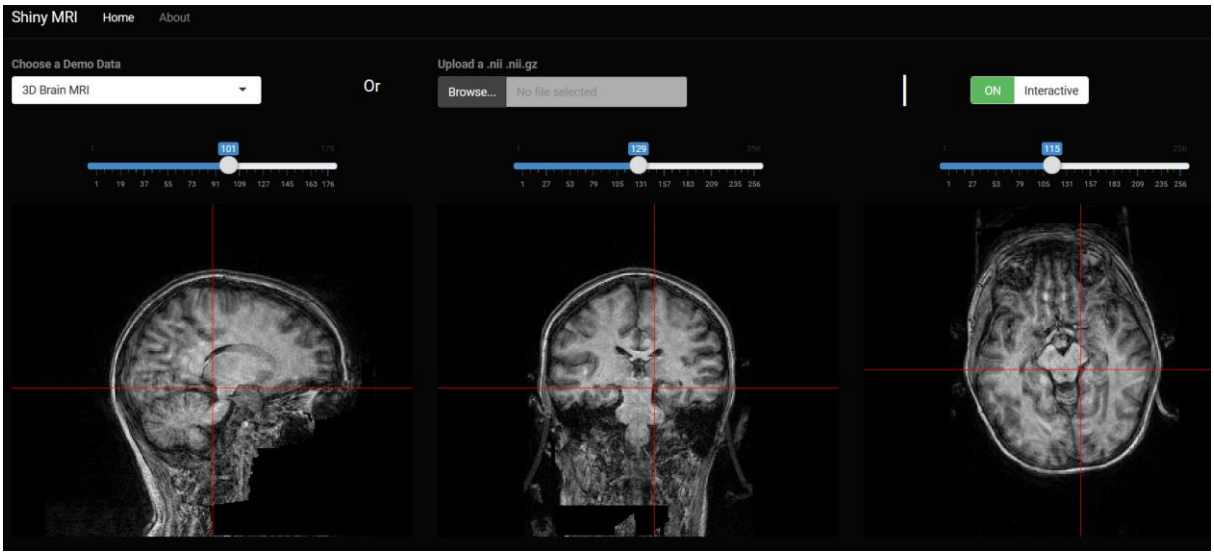
[3] JIA, Lihua et al., 2021. Development of interactive biological web applications with R/Shiny. *Briefings in Bioinformatics.* 23(1) [cit. 06. 02.2025]. Dostupné z: https://doi.org/10.1093/bib/bbab415

činnosti v konkrétnych oblastiach. Tento dashboard je užitočný pre verejnú správu a bezpečnostné zložky, pretože umožňuje identifikovať rizikové oblasti a podporiť efektívne rozhodovanie pri rozmiestnení bezpečnostných síl.



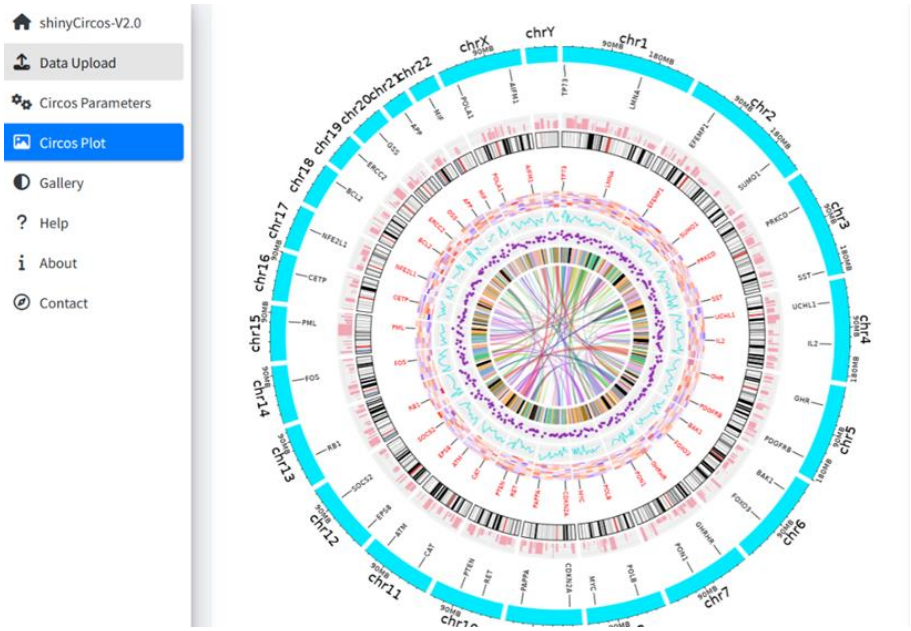
Obr. 1.1.2 Watch Dashboard aplikácia
Zdroj: https://cultureofinsight.shinyapps.io/crime-watch/

- **Shiny MRI** – inovatívny nástroj na vizualizáciu trojrozmerných MRI snímok, používatelia si môžu dynamicky prezerať MRI dáta, čo môže výrazne pomôcť v klinickom výskume a medicínskej diagnostike.



Obr. 1.1.3 Shiny MRI aplikácia
Zdroj: https://haozhu233.shinyapps.io/shinyMRI-contest/

- **shinyCircos** – aplikácia určená na tvorbu Circos diagramov, ktoré sú široko používané v bioinformatike. Vizualizácia komplexných genetických interakcií umožňuje vedcom analyzovať vzťahy medzi rôznymi génmi a chromozómami. Táto aplikácia bola použitá aj v publikáciách v prestížnych vedeckých časopisoch ako Nature, Science a Cell.[4]



Obr. 1.1.4 shinyCircos aplikácia
Zdroj: https://venyao.xyz/shinycircos/

Tieto príklady ukazujú širokú škálu využitia Shiny aplikácií v rôznych oblastiach, od medicíny, ekológie, až po financie. Shiny ponúka flexibilitu pri tvorbe aplikácií, ktoré užívateľom umožnia manipulovať s dátami, meniť parametre vizualizácie alebo vykonávať analýzy bez toho, aby museli čakať na nový výpočet alebo načítanie stránky.

1.1.2 Reaktívne programovanie

Základy reaktívneho programovania siahajú až do 80. rokov 20. storočia, kedy sa táto myšlienka objavila v tabuľkovom editore VisiCalc. Tabuľkové editory využívajú vo veľkej miere princíp reaktivity, keďže užívateľ deklaruje pomocou vzorcov vzťahy, a ak sa jedna bunka zmení, všetky jej závislosti sa automaticky aktualizujú. Myšlienky reaktivity boli spracované

[4] POSIT, PBC. 2024. *Shiny Gallery* [online]. [cit. 06. 02. 2025] Dostupné z: https://shiny.posit.co/r/gallery/

v akademickej informatike koncom 90. rokov a až o dekádu neskôr sa dostali do praktického používania. Priekopnícke frameworky ako Knockout, Ember, a Meteor demonštrovali ako dramaticky jednoduchšie môže byť UI s využitím reaktívneho programovania. Tieto technológie boli inšpiráciou aj pri tvorbe Shiny.

Reaktívne programovanie je kľúčový prístup pre Shiny, keďže jeho hlavnou charakteristikou je interaktivita. Používatelia prostredníctvom vstupných ovládacích prvkov, ako sú stupnice, textové polia, či zaškrtávacie polia, spúšťajú logiku servera, ktorá vedie k aktualizácii výstupov (prekreslenie grafov, aktualizácia tabuliek, atď.).

Tradičné premenné v R umožňujú meniť hodnoty, ale nedokážu zabezpečiť ich automatickú aktualizáciu v prípadoch, keď sa zmení závislá hodnota. Napríklad pri prepočte teploty z Celzia na Fahrenheit by aktualizácia vstupnej premennej `temp_c` nemala žiadny vplyv na výstupnú premennú `temp_f`. Podobne, funkcie v R síce môžu zabezpečiť aktualizáciu výstupu na základe aktuálneho vstupu, no každý výpočet sa vykonáva opakovane, aj keď sa vstupná hodnota nezmenila, čo je neefektívne. Reaktívne programovanie tieto nedostatky rieši pomocou reaktívnych hodnôt `reactiveVal()` a reaktívnych výrazov `reactive()`. Tieto objekty v Shiny automaticky sledujú závislosti medzi vstupmi a výstupmi. Ak dôjde k zmene vstupnej hodnoty, aktualizuje sa iba príslušná časť výstupu, čo optimalizuje výpočty.[5]

1.1.3 Rozširujúce knižnice pre Shiny

Rozvoj platformy Shiny neostal len pri základných funkciách, s ktorými začínala. Tak ako aj v bežnom jazyku R, aj pri zameraní na Shiny komunita programátorov neustále vytvárala nové knižnice, ktoré obohacujú základnú funkcionalitu. Tieto knižnice dnešným používateľom umožňujú vytvárať aplikácie, ktoré sú prepracované ako esteticky, tak aj výkonovo.

Z veľkého množstva knižníc je dôležité spomenúť najmä:

- `shinydashboard` – umožňuje vytvárať dashboardy s využitím preddefinovaných komponentov[6],

[5] WICKHAM, Hadley. *Mastering Shiny*. Why reactivity? [online]. O'Reilly Media 2021 [cit. 24. 01. 2025]. Dostupné z: https://mastering-shiny.org/reactive-motivation.html
[6] RSTUDIO. *shinydashboard: Create Dashboards with 'Shiny'* [online]. The Comprehensive R Archive Network (CRAN)[cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/shinydashboard/

- `shinyjs` – umožňuje integráciu JavaScript operácií bez potreby znalosti JavaScript-u[7],
- `shinyWidgets` – rozširuje Shiny o vylepšené komponenty pre zvýšenie estetiky a uživateľského komfortu[8],
- `plotly` – integruje interaktívne grafy[9],
- `DT` – poskytuje nástroje na vytváranie interaktívnych dátových tabuliek[10],
- `leaflet` – umožňuje integráciu interaktívnych máp s možnosťou pridávania vrstiev, značiek, a ďalších prvkov[11].

1.1.4 Možnosti zdieľania Shiny aplikácií

Tradičné zdieľanie Shiny aplikácií si vyžaduje server, ktorý nepretržite beží, je pripojený na internet, a má nainštalovaný R alebo Python. Takéto zdieľanie je naturálne spojené s nákladmi na hosting, ako aj správu a údržbu serverov. Potreba servera pri zdieľaní môže byť prekážkou pre ľudí, ktorí sa zameriavajú na jednoduchšie a menšie koncepty, v resp. aplikácie.

Moderné technológie prinášajú nové možnosti, ako tento problém riešiť. Jednou z nich je WebAssembly, ktorá umožňuje spúšťať kód lokálne bez závislosti od servera. Ide o nízkoúrovňový jazyk optimalizovaný pre prehliadače, ktorý umožňuje kompiláciu rôznych programovacích jazykov, ako sú C/C++ či Rust, na formát kompatibilný s modernými platformami. Vďaka tejto flexibilite sa niektoré tradičné serverové riešenia mohli presunúť priamo do prehliadača.

---

[7] ATTALI, Dean. *shinyjs: Easily Improve the User Experience of Your 'Shiny' Apps* [online]. The Comprehensive R Archive Network (CRAN) [cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/shinyjs/index.html

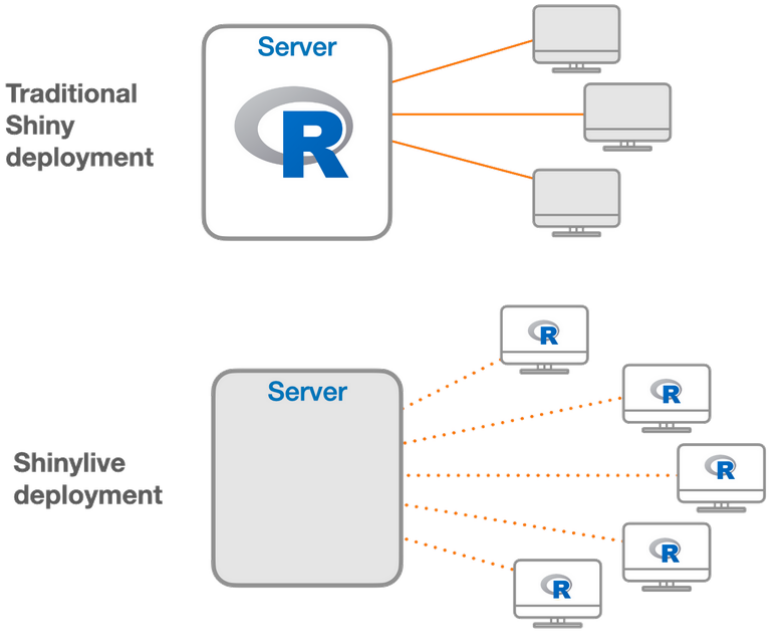
[8] DREAMRS. *shinyWidgets: Custom Inputs Widgets for Shiny Applications* [online]. GitHub [cit. 25. 01. 2025]. Dostupné z: https://github.com/dreamRs/shinyWidgets

[9] POSIT. *Plotly Output for Shiny for Python* [online]. [cit. 25. 01. 2025] Dostupné z: https://shiny.posit.co/py/components/outputs/plot-plotly/

[10] RSTUDIO. *DT: A Wrapper of the JavaScript Library 'DataTables'* [online]. [cit. 25. 01. 2025] Dostupné z: https://rstudio.github.io/DT/

[11] CHENG, Joe; XIE, Yihui; WICKHAM, Hadley. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library* [online]. The Comprehensive R Archive Network (CRAN)[cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/leaflet/index.html

V kontexte Shiny bol tento prístup implementovaný prostredníctvom projektu *Shinylive*. *Shinylive* je založený na kombinácii WebAssembly a nástroja Emscripten, ktorý umožňuje previesť potrebné komponenty R alebo Python na WebAssembly a spustiť ich v prostredí prehliadača.



Obr. 1.1.5 Porovnanie tradičného zdieľania Shiny a zdieľania pomocou Shinylive
Zdroj: https://jcheng5.github.io/posit-conf-2023-shinylive/images/shinylive-shiny-deployment-model-r.png

Shinylive poskytuje tri hlavné spôsoby využitia:

- **Konverzia Shiny aplikácií**: Lokálna aplikácia Shiny (napísaná v R alebo Python) môže byť skonvertovaná do statického HTML, CSS, JavaScriptu a WebAssembly pomocou nástroja *Shinylive*. Týmto spôsobom je možné aplikáciu zdieľať vo forme jednoduchého webového súboru.
- **Priame vytváranie aplikácií v prehliadači**: Platforma Shinylive.io umožňuje používateľom písať a spúšťať Shiny aplikácie priamo v prehliadači, bez nutnosti inštalovať ďalší softvér. Výsledné aplikácie môžu byť zdieľané pomocou URL alebo uložené na GitHub.
- **Integrácia do dokumentov Quarto**: Prostredníctvom Quarto rozšírenia pre *Shinylive* je možné vkladať interaktívne Shiny aplikácie priamo do dokumentov, prezentácií alebo webových stránok.

Medzi výhody takéhoto spôsobu zdieľania je nielen úspora nákladov, ale aj nové možnosti využitia Shiny aplikácií. Umožňuje napríklad jednoduché začlenenie interaktívnych príkladov do prednášok a výučbových materiálov. Avšak má aj svoje obmedzenia, medzi ktoré patrí pomalší čas spúšťania aplikácií, nedostupnosť všetkých balíkov z CRAN a PyPI, ako aj obmedzené možnosti pripájania priamo k databázam.[12]

1.1.5 Shiny pre R vs. pre Python

Balík Shiny bol pôvodne vyvinutý pre jazyk R, no v súčasnosti je dostupný aj pre jazyk Python pod názvom *PyShiny*. Táto verzia umožňuje širšej komunite dátových analytikov využívať výhody interaktívnych aplikácií bez nutnosti meniť svoj preferovaný jazyk.

Napriek podobnej funkcionalite existujú medzi týmito verziami určité rozdiely, ktoré môžu ovplyvniť výber jazyka:

- **Základný kód**: R Shiny aj Shiny pre Python vyžadujú minimálny začiatočný kód pre spustenie a syntax je takmer identická. R Shiny používa jednoduchšiu, menej modulárnu štruktúru importov, avšak ani jeden prístup nie je lepší, záleží skôr na osobných preferenciách.
- **UI komponenty**: Kľúčový rozdiel medzi R Shiny a *PyShiny* je v spôsobe pomenovania komponentov. Prístup Python-u je viac konzistentný (všetky vstupné prvky začínajú prefixom `input_`), zatiaľ čo v R je slovo *input* umiestené na konci názvu funkcie. Pre začiatočníkov môže byť Python jednoduchší na navigáciu.
- **Serverová logika**: Viditeľným rozdielom je spôsob volania prvkov, pri R Shiny má každý prvok svoju korešpondenčnú funkciu `render*()`, zatiaľ čo PyShiny používa dekorátory `@output` a `@render.<prvok>`.

Rozdiely medzi týmito nástrojmi sú minimálne, pričom oba sú skvelé na tvorbu interaktívnych dashboardov a aplikácií, voľba jazyka závisí od programátorských zručností a požiadaviek projektu.[13]

---

[12] CHENG, Joe. *Running Shiny without a server* [online]. GitHub, 2023 [cit. 24. 01. 2025]. Dostupné z: https://jcheng5.github.io/posit-conf-2023-shinylive/?print-pdf=#/title-slide

[13] RADEČIĆ, Dario. *R Shiny vs Shiny for Python: What are the Key Differences* [online]. Appsilon, 2022 [cit. 25. 01. 2025]. Dostupné z: https://www.appsilon.com/post/r-shiny-vs-shiny-for-python

1.2 Výhody a nevýhody R Shiny v porovnaní s inými nástrojmi

1.2.1 Nástroje Business Intelligence

V súvislosti s vizualizáciou dát a vytváraním interaktívnych dashboardov siahajú firmy často po nástrojoch business intelligence, ako sú Power BI, Tableau, či Spotfire. Tieto nástroje ponúkajú jednoduché riešenia pre spracovanie a prezentáciu dát, no otázkou zostáva, či dokážu konkurovať riešeniam, ktoré umožňujú väčšiu mieru prispôsobenia, akým je R Shiny. V tab. 1.2.1 si tieto nástroje porovnáme z pohľadu flexibility, analytických schopností, výkonu a nákladov.

Tab. 1.2.1 Porovnanie R Shiny s nástrojmi Business Intelligence

Kategória	R Shiny	Power BI	Spotfire	Tableau
Flexibilita a prispôsobenie	R Shiny je vo veľkej miere prispôsobiteľný, keďže umožňuje vytvárať aplikácie šité na mieru. Táto flexibilita môže viesť k nižším nákladom, ak má organizácia vlastných skúsených R programátorov.	Power BI je prienikom jednoduchosti používania a prispôsobenosti, keďže ponúka užívateľsky prívetivé rozhranie. Pokročilé požiadavky na prispôsobenie môžu vyžadovať vyššie náklady na prémiové analytické funkcie.	Spotfire poskytuje všestranné nástroje pre pokročilú vizualizáciu a prispôsobenie údajov. Avšak čím komplexnejšie úlohy, tým väčšia pravdepodobnosť väčších nákladov.	Tableau ponúka veľké možnosti prispôsobenia prostredníctvom drag-and-drop rozhrania. Avšak pri zložitejších požiadavkách sú potrebné znalosti Tableau skriptu.
Pokročilá analytika a vizualizácia	R Shiny, vychádzajúci z jazyka R, má veľkú výhodu v oblasti štatistiky a pokročilej analýzy. Ponúka rozsiahlu flexibilitu pri analytických problémoch, pričom je veľmi všestranný pre potreby špecializovaných údajov.	Ak je integrovaný s Microsoft, môže byť široko využívaný v oblasti analytiky. Je ideálny pre podniky s analytickými a vizuálnymi požiadavkami.	Spotfire vyniká svojou pokročilou analýzou vrátane prediktívneho modelovania a dolovania údajov. Je to častá voľba firiem vyžadujúcich hlbšiu analytickú zdatnosť a sofistikovanú vizualizáciu.	Tableau vyniká vo vizualizácii údajov so širokou škálou vopred vytvorených vizuálnych prvkov. Je však primárne zameraný skôr na vizualizáciu než hĺbkovú štatistickú analýzu.

Použiteľnosť pre netechnických používateľov	R Shiny vyžaduje znalosti programovania, avšak po jeho osvojení poskytuje rozmanitosť, ktorá vyhovuje technickým aj netechnickým požiadavkám.	Power BI je známky svojou jednoduchosťou, čo vyhovuje ľudom s minimálnymi technickými znalosťami. Jeho rozhranie drag-and-drop ho robí intuitívnym a prístupným pre široké spektrum používateľov.	Spotfire ponúka intuitívne používateľské rozhranie, vďaka čomu je prístupné technickým aj netechnickým používateľom. Jeho jednoduché používanie je porovnateľné s Power BI, no s väčšou škálou pokročilých nástrojov.	Tableau je veľmi intuitívny a určený pre netechnických používateľov. Pokročilá analýza ale vyžaduje podporu technicky zdatných osôb.
Škálovateľnosť výkon a rýchlosť vývoja	Shiny je vysoko škálovateľný a funguje dobre, keď je optimalizovaný. Vývoj aplikácií však môže byť pomalší kvôli potrebe odborných znalostí R a manuálnej úprave.	Power BI je škálovateľný, najmä v prostredí Microsoftu. Pri znalosti Microsoftu ponúka vysoký výkon a vývoj, keďže vie rýchlo prepájať projekty v rámci rovnakého prostredia.	Spotfire je škálovateľný pre podniky všetkých veľkostí a ponúka dobrý výkon. Rýchlosť vývoja je zvyčajne vyššia vďaka intuitívnemu rozhraniu a vopred vytvoreným šablónam pre rôzne prípady použitia.	Tableau je vysoko škálovateľný a funguje skvelo v podnikových prostrediach. Vývoj prebieha veľmi rýchlo, najmä pre skúsených používateľov. Vývoj je pomalší pri veľkých projektoch alebo projektoch vyžadujúcich vysoké prispôsobenie.
Nákladová efektívnosť	Shiny funguje ako open-source, čo ho robí nákladovo najefektívnejším, najmä pre organizácie so skúsenými R programátormi. Môžu sa vytvoriť náklady na hosting a prevádzku, avšak aspoň sa neplatia žiadne licenčné poplatky.	Power BI ponúka nízke ceny, najmä pre podniky, ktoré už využívajú Microsoft. Jeho predplatné spadá do nákladovo efektívnej úrovne, avšak prémiové funkcie môžu tieto náklady zvýšiť.	Spotfire môže byť nákladný, najmä pre malé a stredne veľké organizácie. Aj keď poskytuje vynikajúce funkcie, z dlhodobého hľadiska je pre tímy s menším rozpočtom skôr cenovo nedostupný.	Tableau je prémiový produkt s náležitými licenčnými nákladmi. Je vhodný pre organizácie, ktoré sú ochotné investovať do kvalitných vizualizačných nástrojov. Pre menšie tímy alebo jednorazové projekty nie je nákladovo efektívny.
Náklady na údržbu	Údržba aplikácií vyžaduje pravidelné aktualizácie prostredia a balíkov R. Shiny ako open-source môže znamenať, že aktualizácie a opravy chýb sú vo vlastnej réžii firmy, čo môže znamenať ďalšie náklady v závislosti od zložitosti aplikácií.	Power BI má zjednodušený proces aktualizácie a údržby ako súčasť ekosystému spoločnosti Microsoft. Licenčné poplatky a predplatenie prémiových funkcií však môže zvýšiť náklady na dlhodobú údržbu.	Spotfire je prémiový produkt, ktorý poskytuje spoľahlivé fungovanie a pravidelné aktualizácie. Náklady sú ale všeobecne vyššie v porovnaní s open-source nástrojmi ako je napríklad Shiny.	Tableau ponúka pravidelné aktualizácie ako súčasť svojho licenčného modelu. Náklady na údržbu môžu byť relatívne vysoké, najmä pri lokálnych inštaláciách alebo pri správe v rozsiahlych organizáciách. Súčasťou je však silná zákaznícka podpora.

Zdroj: vlastné spracovanie podľa https://www.appsilon.com/post/rshiny-vs-powerbi-vs-spotfire

R Shiny ponúka veľkú flexibilitu a zameranie na štatistiku, pričom jeho výhodou je možnosť prispôsobiť aplikácie do najmenších detailov. Na druhej strane, BI nástroje, vynikajú intuitívnym prostredím, vďaka čomu sú prístupnejšie aj pre menej technicky zdatných používateľov. Konečné rozhodnutie závisí od konkrétnych požiadaviek firmy, dostupného rozpočtu a technických zručností tímu.

1.2.2 Dash v jazyku Python vs. Shiny v R

V oblasti interaktívnej vizualizácie dát patria medzi najvyužívanejšie nástroje najmä frameworky Dash a R Shiny, pričom každý z nich ponúka špecifické výhody v závislosti od potrieb používateľov a zvoleného programovacieho jazyka. Dash bol vyvinutý spoločnosťou Plotly v roku 2017 ako open-source framework orientovaný na vývoj komplexných dashboardov s dôrazom na firemné prostredie a veľké dátové toky. Na rozdiel od Shiny, ktorá má silný akademický a výskumný pôvod, Dash sa využíva najmä v komerčnej sfére – v odvetviach ako financie, telekomunikácie, či priemysel, kde je kľúčová schopnosť rýchlo vizualizovať veľké objemy dát a zároveň udržať vysoký štandard používateľského rozhrania.[14] Ich využitie však nie je jediný z ich rozdielov.

Výhody Dash v Python:

- *Práca v populárnom jazyku:* Dash je navrhnutý pre Python, ktorý patrí medzi najpopulárnejšie programovacie jazyky. Python má v oblasti dátovej vedy niekoľko známych knižníc, ako napríklad `pandas` alebo `numpy`.
- *Flexibilita:* Dash umožňuje tvorbu aplikácií s viacerými stránkami a dynamickými navigačnými prvkami.
- *Deklaratívny prístup.*

Nevýhody Dash v Python:

- *Zložitejšie vizualizácie*: V porovnaní so Shiny vyžaduje viac manuálnej práce pri nastavení a dizajne prvkov.

[14] CASTILLO, Dylan. *Develop Data Visualization Interfaces in Python With Dash* [online]. Real Python, 2025 [cit. 18. 04. 2025]. Dostupné z: https://realpython.com/python-dash/

- *Menej špecializovaný na štatistiku:* Python síce podporuje štatistickú analýzu, no Shiny má v tejto oblasti miernu výhodu vďaka integrácii s R.
- *Náročný pre začiatočníkov:* Pre úplných začiatočníkov môže byť deklaratívny štýl a jeho integrácia náročná na pochopenie.

Výhody R Shiny:

- *Osvedčenosť:* Shiny je na trhu od roku 2012, má širokú komunitu, bohatú dokumentáciu, a množstvo knižníc pre uľahčenie tvorby aplikácií.
- *Štatistická analýza:* Shiny je predovšetkým integrovaný s jazykom R, ktorý je celosvetovo uznávaný jazyk pre štatistiku a dátovú analýzu.
- *Množstvo a jednoduchosť knižníc:* Sú podporované všetky knižnice, ktoré podporuje samotný jazyk R.

Nevýhody R Shiny:

- *Závislosť na R:* R, napriek svojím prednostiam v štatistike, nie je univerzálny programovací jazyk a môže byť menej známy v svete programátorov.
- *Výkonnostné obmedzenia:* Komplexné aplikácie musia byť optimalizované, aby fungovali správne.
- *Náročný pre začiatočníkov:* Pre nových užívateľov môže byť tvorba aplikácií v Shiny náročná, najmä bez predchádzajúcich skúseností s R.[15]

Z porovnania vyplýva, že voľba by mala zohľadňovať cieľovú oblasť a jazykovú preferenciu vývojára – pre rozsiahle podnikové dashboardy s externými službami je vhodnejší Dash v jazyku Python, a pre špecializované štatistické či vedecké aplikácie je vhodnejší jazyk R a balík Shiny.

1.3 Možnosti využitia Shiny v úlohách Management Science

Management Science je interdisciplinárna veda zaoberajúca sa vývojom a aplikáciou modelov a konceptov, ktoré pomáhajú pri manažérskych rozhodovacích procesoch. Vo svojej

[15] RADEČIĆ, Dario. *Python Dash vs. R Shiny – Which to Choose in 2021 and Beyond* [online]. Appsilon, 2020 [cit. 26. 01. 2025]. Dostupné z: https://www.appsilon.com/post/dash-vs-shiny

podstate vychádza z operačného výskumu, prelína sa však s disciplínami z inžinierstva, dátovej vedy, teórii hier, krízového manažmentu, a pod.

Táto vedná disciplína sa vo svojich začiatkoch zameriavala na akúkoľvek aplikáciu vedy na riešenie manažérskych problémov a samotný proces manažmentu. Do jej rozsahu spadali oblasti ako operačný výskum, systémová analýza či štúdium manažérsko-informačných systémov. Toto široké ponímanie viedlo v roku 1953 k založeniu *Institute of Management Sciences* na podnet *Operations Research Society of America*, pričom jeho hlavným cieľom bolo identifikovať, zhromažďovať a integrovať poznatky relevantné pre manažérsku prax.

V súčasnosti je **Management Science** chápaná ako koncept, ktorý sa zameriava na tvorbu a aplikáciu modelov a analytických metód na riešenie problémov manažmentu. Využíva predovšetkým metódy analýzy dát, štatistické prístupy a optimalizačné techniky na zlepšenie efektivity manažmentu. Keďže manažment je v tomto prístupe vnímaný ako logický proces, je možné ho vizualizovať, algoritmizovať, kvantifikovať a popísať pomocou symbolov, meraní a vzťahov.

Tento systematický prístup čerpá z teórie rozhodovania a racionálnych modelov rozhodovania, pričom organizáciám poskytuje nástroje na identifikáciu cieľov a efektívne plánovanie krokov na ich dosiahnutie. **Management Science** tak pomáha identifikovať a riešiť kľúčové problémy organizácie, zvyšovať efektivitu riadenia, optimalizovať využitie zdrojov a vytvárať strategické plány na dosiahnutie stanovených cieľov.[16]

Medzi typické problémy, ktorými sa zaoberá sú:

- Alokácia zdrojov.
- Logistika.
- Prieskum trhu.
- Skladovanie zásob.
- Investičné rozhodnutia.
- Problémy v oblasti výroby.
- Výber produktového mixu.
- Rozloženie ľudských zdrojov.

[16] BREZINA, Ivan a PEKÁR, Juraj. *Management Science: riešenie optimalizačných úloh prostredníctvom MS EXCEL a jazyka R*. I. Vydanie prvé. Bratislava: Ekonóm, 2024, s. 7 . ISBN 978-80-225-5140-3.

- Analýza efektívnosti, a ďalšie.

Tieto problémy potom možno riešiť napríklad pomocou matematického programovania, štatistických metód, sieťových modelov, metódami viackriteriálneho rozhodovania, a i.[17]

Riešenie rozhodovacích úloh závisí od štruktúrovanosti problému a jeho výpočtovej náročnosti. Pre štruktúrované úlohy s nízkou zložitosťou sú efektívne exaktné metódy, ktoré zaručujú optimálne riešenie. Ak exaktné metódy nie sú použiteľné, využívajú sa heuristiky alebo metódy umelej inteligencie. Heuristiky nezaručujú optimálne riešenie, ale poskytujú rýchle suboptimálne riešenie pre úlohy s vysokou výpočtovou náročnosťou.[18]

Tab. 1.3.1 Klasifikácia metód na riešenie optimalizačných rozhodovacích úloh

Typ metódy	Typ úlohy	Riešenie	Výpočtová zložitosť
Exaktná	Dostatočne štruktúrovaná	Optimálne riešenie, ak existuje	Nízka, efektívne pre malé úlohy
Heuristická	Nedostatočne štruktúrovaná	Suboptimálne riešenie, rýchle, ale nie zaručene optimálne	Vysoká, vhodné pre komplexné úlohy
Kombinovaná	Rôznorodé podúlohy	Kombinácia exaktných a heuristických prístupov	Závisí od kombinácie metód a podúloh

Zdroj: vlastné spracovanie podľa Brezina, I. a Pekár J. *Management Science*. 2024, s. 8.

1.3.1 Modelovanie podnikových procesov

Pri analýze podnikových systémov nie je potrebné spracovať detaily o všetkých ich prvkoch, ale skôr vymedziť hlavné aspekty a vzťahy. Možno ich potom skúmať pomocou modelu, ktorý odzrkadľuje zjednodušený odraz reality podnikového procesu.

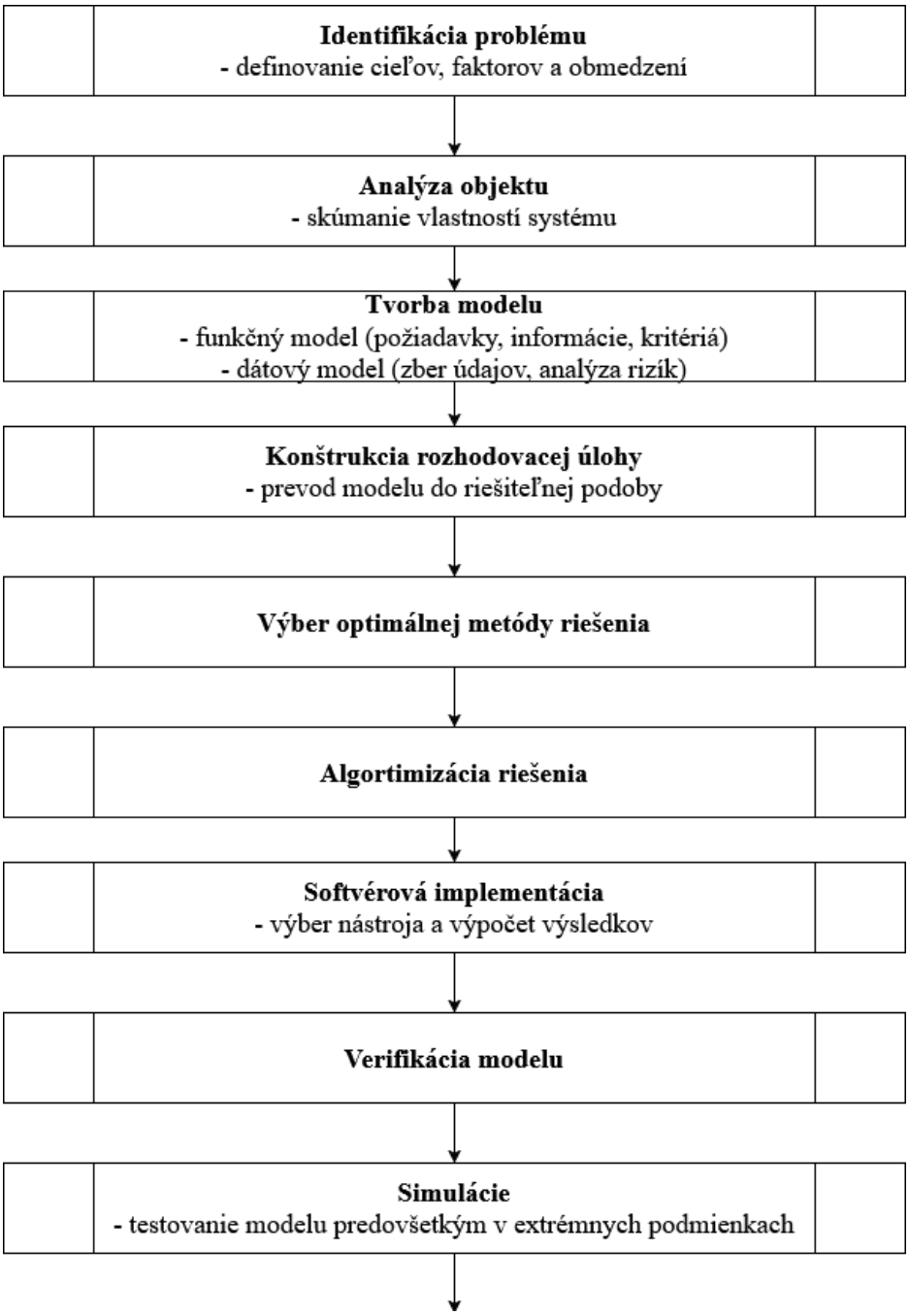
Ekonomicko-matematické modely popisujú procesy pomocou matematických vzťahov, pričom modelované veličiny možno vyjadriť ako premenné, parametre a konštanty. Cieľom modelovania je vytvoriť riešiteľné modely, ktoré umožňujú optimalizáciu a podporu rozhodovania pomocou vhodných matematických metód.

Často sa využívajú klasické matematické metódy, avšak pri riešení úloh z praxe môžu modely byť veľmi rozsiahle a vyžadovať veľkú výpočtovú zložitosť, čo predstavuje významnú

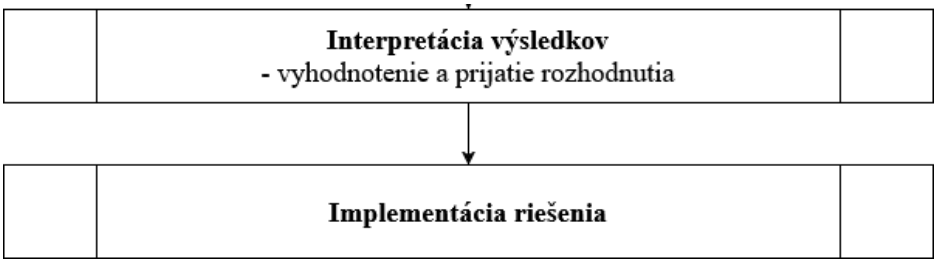
[17] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 8.
[18] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 13.

prekážku. Preto sa čoraz viac využívajú netradičné prístupy, najmä z oblasti umelej inteligencie.[19]

Proces modelovania možno rozdeliť do nasledujúcich krokov, ktoré sú znázornené v schéme:



[19] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 11.



Obr. 1.3.1 Schéma postupu modelovania.
Zdroj: vlastné spracovanie podľa Brezina, I. a Pekár J. *Management Science*. 2024, s. *12*

1.3.2 Optimalizačné úlohy

Matematické programovanie sa zaoberá modelovaním akéhokoľvek systému a jeho optimalizáciou. Cieľom je na základe určitého kritéria nájsť extrém, ak existuje, pri určitých podmienkach. Takýto cieľ sa vyjadruje prostredníctvom *účelovej funkcie*.[20]

Ak sú všetky vzťahy v úlohe matematického programovania lineárne, hovoríme o **lineárnom programovaní**. Riešenie úloh pomocou lineárneho programovania je vo svojej matematickej štruktúre pomerne jednoduché, a zároveň využiteľné na riešenie mnohých úloh.

Vo všeobecnom tvare možno model lineárneho programovania zapísať:

$$f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=i}^{n} a_{ij} x_j \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i$$

$$x_j \geq 0$$

kde: c_j – koeficienty účelovej funkcie, $j = 1,2, ...n$,

a_{ij} – technologické koeficienty sústavy ohraničení, $i = 1,2, ...,m, j = 1,2, ...n$,

b_i – koeficienty pravej strany, $i = 1,2, ...m$,

x_j – rozhodovacie premenné, $j = 1,2, ...n$.

[20] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 66.

Taktiež je možné model zapísať v maticovom tvare:

$$f(\mathbf{x}) = \mathbf{c^T x}$$

$$\mathbf{A}x \leq \mathbf{b}$$

$$\mathbf{x} \geq 0$$

kde: $\mathbf{c^T}$ – vektor rozmeru *1*x*n* koeficientov účelovej funkcie c_j, j = 1,2, ...*n*,

A – matica rozmeru *m*x*n* technologických koeficientov a_{ij}, i = 1,2, ...,*m*, j = 1,2, ...*n*,

b – vektor rozmeru *m*x*1* koeficientov pravej strany b_i, i = 1,2, ...*m*,

x – vektor rozmeru *1*x*n* rozhodovacích premenných x_j, j = 1,2, ...*n*,

0 – vektor rozmeru *m*x*1* obsahujúci hodnoty 0.[21]

Ako typické modely lineárneho programovania možno uviesť optimalizáciu výrobného procesu, rezný problém, dopravnú úlohu, priraďovací problém, a. i.

Medzi najznámejšie a najčastejšie riešené problémy v rámci optimalizácie prepravy patrí *hľadanie najkratšej cesty* alebo *hľadanie najkratšej okružnej cesty*, napríklad medzi odberateľom a dodávateľom. Riešenie týchto problémov je založené na teórii grafov, kde graf reprezentuje priestorovú štruktúru s časovými väzbami medzi jednotlivými subjektmi.[22]

1.3.3 Softvérové možnosti riešenia optimalizačných úloh

V programe **MS Excel** je možné riešiť úlohy **matematického programovania**, vrátane úloh **lineárneho programovania** (ÚLP), pomocou doplnku *Solver* (*Riešiteľ*). Tento doplnok však musí byť najskôr aktivovaný v nastaveniach programu. Vstupné údaje pre riešenie ÚLP môžu byť usporiadané ľubovoľne, avšak je nevyhnutné dodržať pravidlá požadované *Riešiteľom*.

Na nasledujúcom príklade si ukážeme postup riešenia ÚLP v MS Excel:

[21] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 69.
[22] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 168.

Pekáreň chce optimalizovať výrobu ražného a pšeničného chleba. Každý chlieb vyžaduje múku a kvasnice:

- Ražný chlieb potrebuje 3 kg múky a 1 balenie kvasníc na kus.
- Pšeničný chlieb potrebuje 2 kg múky a 2 balenia kvasníc na kus.

Pekáreň má k dispozícii 30 kg múky a 20 balení kvasníc. Cieľom je maximalizovať zisk, pričom zisk na jednom ražnom chlebe predstavuje 5 € a pšeničnom 4 €.

Tento príklad by sme formulovali ako ÚLP nasledovane:

Maximalizujeme: $z(\mathbf{x}) = 5x_1 + 4x_2$

Obmedzenia: $3x_1 + 2x_2 \leq 30$

$x_1 + 2x_2 \leq 20$

$x_1, x_2 \geq 0$

Vstupné údaje predstavujú:

- **Koeficienty účelovej funkcie** (c_j): očakávaný zisk, t. j. 5 € a 4 €,
- **Štrukturálne koeficienty** (a_{ij}): vyjadrujú koľko surovín potrebujeme na výrobu jednotlivých druhov chleba,
- **Disponibilné zdroje** (b_i): 30 kg múky a 20 balení kvasníc.

Výsledkom výpočtu majú byť:

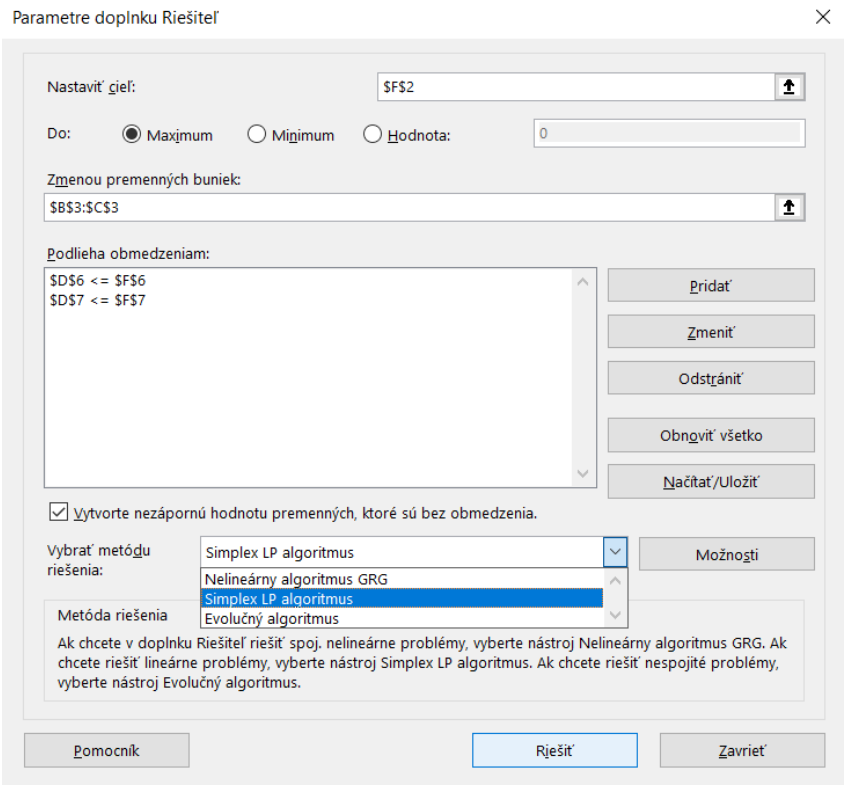
- **Hodnoty rozhodovacích premenných** (x_j): počet vyrobených chlebov,
- **Hodnota účelovej funkcie** ($f(\mathbf{x})$): celkový zisk.

	A	B	C	D	E	F
1		x_1	x_2			
2	koeficienty ÚF	5	4		ÚF	=SUMPRODUCT(B2:C2;B3:C3)
3	rozhodovacie *xj*	0	0			
4						
5	obmedzenia					b_i
6	múka	3	2	=SUMPRODUCT(B6:C6;$B$3:$C$3)	<=	30
7	kvasnice	1	2	=SUMPRODUCT(B7:C7;$B$3:$C$3)	<=	20

Obr. 1.3.2 Príprava údajov pre riešenie ÚLP pomocou doplnku Riešiteľ

Zdroj: vlastné spracovanie

Na začiatku sa vstupné hodnoty rozhodovacích premenných nastavujú na 0. Optimalizačné kritérium sa zapisuje vo forme vzorca, ktorý predstavuje skalárny súčin vektora očakávaného zisku a vektora rozhodovacích premenných. V MS Excel na tento výpočet slúži funkcia =*SUMPRODUCT(oblasť1, oblasť2)*, kde *oblasť1* obsahuje koeficienty účelovej funkcie a *oblasť2* rozhodovacie premenné.



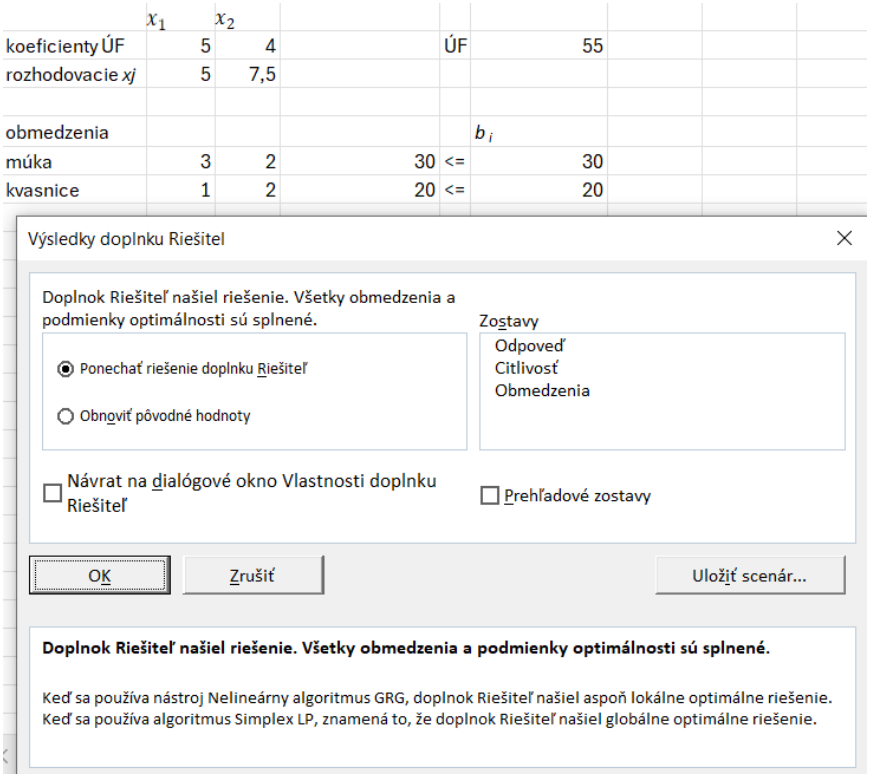
Obr. 1.3.3 Dialógové okno *Parametre doplnku Riešiteľ*
Zdroj: vlastné spracovanie

Po príprave vstupných údajov sa aktivuje *Riešiteľ*, kde upravujeme parametre modelu pomocou dialógového okna (Obr. 1.3.3) :

- *Nastaviť cieľ* – určuje bunku s účelovou funkciou, ktorej hodnotu optimalizujeme (F2),
- *Do* – definuje typ optimalizácie, k dispozícii máme maximalizáciu, minimalizáciu alebo dosadenie cieľovej hodnoty (náš príklad vyžaduje maximalizáciu),
- *Zmenou premenných buniek* – vyberieme bunky, ktoré sa budú meniť a na konci výpočtu budú obsahovať výsledné hodnoty (B3:C3),

- *Podlieha obmedzeniam* – nastavíme podmineky, t. j. odkaz na bunku obsahujúcu vzorec, typ obmedzenia (≤, =, ≥, int, bin, dif), a hodnotu/bunku obmedzenia,
- *Výber metódy riešenia* – v ponuke máme nelineárny algorimus GRG, Simplexov LP algoritmus, alebo Evolučný algoritmus,
- *Prepínač nezáporných hodnôt premenných* – väčšinou zapnutý, aby nemohli premenné nadobúdať záporné hodnoty,
- *Tlačidlo* „Riešiť“ – spustenie výpočtu, pričom môže trvať rôzne dlho v závislosti od veľkosti úlohy a počítačovej výkonnosti.

Po skončení výpočtu sa zobrazí dialógové okno s informáciou, či bolo nájdené optimálne riešenie (Obr. 1.3.4). Používateľ má môžnosť si uchovať vypočítané riešenie alebo vrátiť bunky na pôvodné hodnoty. Okrem základného riešenia je možné vygenerovať aj podrobnejšie zostavy, ktoré sa zobrazia v novom hárku.



Obr. 1.3.4 Dialógové okno *Výsledky doplnku Riešiteľ* a riešenie ÚLP
Zdroj: vlastné spracovanie

Optimálne riešenie nášho príkladu:

$$f(\mathbf{x})^* = x_1^* + x_2^*,$$

kde: $x_1^* = 5$

$x_2^* = 7{,}5$

$f(\mathbf{x})^* = 55$

Slovná interpretácia: Pekárni sa najviac oplatí vyrobiť 5 ks ražného chleba a 7,5 ks pšeničného chleba. Pri tomto výrobnom pláne sa maximalizuje celkový zisk na hodnotu 55 €, pričom sa neprekročia dostupné zásoby disponobilných zdrojov.

Úlohy lineárneho programovania možno riešiť aj v jazyku R pomocou knižnice `lpSolve`. Tento balík poskytuje funkciu `lp()`, ktorá umožňuje definovať a riešiť ÚLP zadaním parametrov, v resp. vektorov **c** a **b**, matice **A** a typov ohraničení jednotlivých štruktúrnych ohraničení.

Na obrázkoch č. 1.3.5 a 1.3.6 je zobrazené riešenie predošlého príkladu pomocou balíka `lpSolve` v jazyku R.

```
library(lpSolve)
c <- c(5,4)
A <- matrix(c(3,2,
              1,2), nrow=2, byrow=TRUE)
b <- c(30,20)
ohr <- c("<=", "<=")
lp("max",c,A,ohr,b)
lp("max",c,A,ohr,b)$solution
```

Obr. 1.3.5 Kód riešenia ÚLP v jazyku R
Zdroj: vlastné spracovanie

```
> lp("max",c,A,ohr,b)
Success: the objective function is 55
> lp("max",c,A,ohr,b)$solution
[1] 5.0 7.5
```

Obr. 1.3.6 Výstup z konzoly R
Zdroj: vlastné spracovanie

Po predstavení riešenia ÚLP v Exceli a jazyku R je vhodné zvážiť aj ďalšie nástroje, ktoré môžu zefektívniť prácu s optimalizačnými modelmi. Tradičné nástroje, ako je MS Excel, R, alebo špecializované softvéry pre operačný výskum, síce poskytujú široké možnosti výpočtov, avšak sú limitované v oblasti interaktivity, vizualizácie, a dynamickej práci so vstupmi a výstupmi. Tieto nedostatky dokáže eliminovať R Shiny, ktorý umožňuje integráciu

modelov do webových aplikácií dostupných širšiemu okruhu používateľov vrátane tých bez technického zázemia.

Dôvody prečo môže byť R Shiny vhodnou alternatívou:

- **Zjednodušenie práce:** Užívatelia môžu interaktívne zadávať parametre a obmedzenia modelov, a okamžite vizualizovať výsledky vo forme grafov, tabuliek, či máp.
- **Dostupnosť:** Intuitívne rozhranie aplikácie sprístupní úlohy pre používateľov bez predchádzajúcich skúseností s programovaním či matematikou.
- **Flexibilita:** Umožňuje integrovať rôzne metódy a algoritmy, od lineárneho programovania až po teóriu grafov či strojového učenia.
- **Možnosť integrácie:** Aplikácie môžu byť prepojené s databázami, inými nástrojmi alebo externými softvérmi, čím sa rozširuje ich použiteľnosť.
- **Automatizácia výpočtov:** Umožňuje opakovane riešiť podobné úlohy bez manuálnych zásahov, čo šetrí čas a minimalizuje riziko chýb.
- **Spolupráca a zdieľanie:** Viacerí používatelia môžu pracovať s rovnakou aplikáciou súčasne, čo uľahčuje tímovú prácu a efektívne zdieľanie výsledkov.

2 CIEĽ PRÁCE

Hlavným cieľom bakalárskej práce je navrhnutie interaktívnej aplikácie s využitím prostredia Shiny v jazyku R a overenie jej praktickej využiteľnosti pri riešení vybraných úloh.

Na dosiahnutie hlavného cieľa je nevyhnutné splniť niekoľko čiastkových úloh:

1. Definovať prostredie Shiny a identifikovať jeho výhody a nevýhody.
2. Vybrať a opísať vhodnú oblasť implementácie aplikácie.
3. Popísať proces tvorby aplikácie a jej funkcionalitu v nami vybranej oblasti, konkrétne v úlohách Management Science.
4. Vyhodnotiť funkcionalitu aplikácie.

Prvé dva čiastkové ciele sú spracované v prvej časti práce, kde sa najprv venujeme teoretickému rámcu Shiny, jeho funkcionalite a histórii, ako aj výhodách či nevýhodách v porovnaní s inými webovými nástrojmi. V práci sme sa rozhodli pre oblasť Management Science, ktorá zahŕňa riešenie rôznych optimalizačných úloh. Teoretický rámec obsahuje základný prehľad tejto disciplíny, typické príklady úloh a tradičné prístupy k ich riešeniu pomocou softvérových nástrojov.

Tretí čiastkový cieľ je podrobne rozpracovaný v nasledujúcej časti práce, kde sa zameriavame na jednotlivé kroky pri vývoji aplikácie s využitím jazyka R a frameworku Shiny. Najprv približujeme štandardné postupy riešenia dopravnej úlohy, výrobného problému, hľadania najkratšej a najkratšej okružnej cesty. Následne predstavujeme štruktúru aplikácie rozdelenú na užívateľské rozhranie a serverovú logiku, opisujeme použité knižnice a konkrétny spôsob implementácie jednotlivých úloh.

Štvrtá úloha je rozpracovaná v záverečnej časti práce, kde analyzujeme výkonnosť vytvorenej aplikácie pomocou praktickej aplikácie na konkrétne príklady, pričom výsledky porovnávame s manuálnymi výpočtami alebo prepočítavame v jazyku R. Nakoniec hodnotíme silné a slabé stránky vytvorenej aplikácie.

3 METODIKA PRÁCE A METÓDY SKÚMANIA

3.1 Všeobecný proces tvorby Shiny aplikácie

Na tvorbu akejkoľvek aplikácie Shiny je potrebné nainštalovať balík pomocou príkazu `install.packages("Shiny")`. Každá aplikácia sa skladá z dvoch hlavných častí: **UI** (user interface), ktorý predstavuje frontend aplikácie, teda definuje vzhľad a rozloženie aplikácie, a **serverovej funkcie**, ktorá je backendom aplikácie, v respektíve spracováva logiku určujúcu správanie aplikácie.

Pri vytváraní aplikácie je potrebné vytvoriť nový priečinok, v ktorom sa uloží skript aplikácie. Tento skript by mal mať názov podľa aplikácie (napr. app.R) a mal by obsahovať kód už vo vyššie spomínanej štruktúre.

```
library(shiny)
ui <- fluidPage(
  "Jednoduchá aplikácia"
)
server <- function(input,output) { }
shinyApp(ui,server)
```

Obr. 3.1.1 Kód jednoduchej Shiny aplikácie
Zdroj: vlastné spracovanie

Aplikáciu je možné spustiť viacerými spôsobmi. V prípade použitia prostredia Rstudio možno kliknúť na tlačidlo Run App v pravom hornom rohu panela s kódom alebo použiť klávesovú skratku Ctrl + Shift + Enter. Po spustení aplikácie si v konzole môžeme všimnúť informácie o pripojení aplikácie k lokálnemu serveru. Tieto informácie obsahujú URL adresu, ktorá slúži na prístup k aplikácii. Po zadaní tejto URL adresy do kompatibilného webového prehliadača vieme otvoriť kópiu našej aplikácie v prehliadači.

```
Listening on http://127.0.0.1:3285
```

Obr. 3.1.2 Informácie o pripojení aplikácie v konzole
Zdroj: vlastné spracovanie

Obr. 3.1.3 Tlačidlo pre spustenie aplikácie
Zdroj: vlastné spracovanie

Počas spustenia aplikácie je R konzola „zablokovaná“, čo znamená, že nie je možné vykonávať ďalšie príkazy, pokým aplikácia neukončí svoju činnosť. Zastaviť ju môžeme opäť niekoľkými spôsobmi, ako napr. stlačením ikony Stop alebo len zatvorením okna aplikácie.

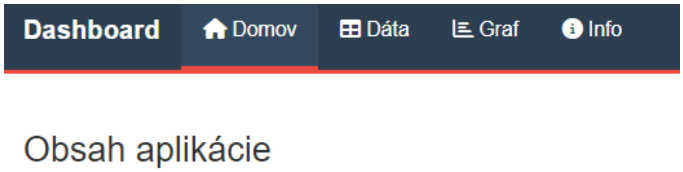
Pri experimentovaní s aplikáciou v rámci prostredia RStudio, nie je potrebné aplikáciu zastavovať a znova spúšťať na zobrazenie zmien, stačí použiť tlačidlo Reload app.

3.1.1 UI (user interface)

Vzhľad Shiny aplikácie je definovaný v používateľskom rozhraní (UI), ktoré určuje celkové rozloženie aplikácie, ovládacie prvky, a zobrazovaný obsah. Je dôležité zvoliť vhodné rozloženie, aby bola aplikácia prehľadná, či už ide o jednoduché zobrazenie informácií alebo komplexnú interaktívnu aplikáciu.

V Shiny existuje niekoľko možností, ako usporiadať obsah:

- **Navigačné panely (navbars)** – môžu byť umiestnené v hornej alebo dolnej časti, prípadne meniť polohu dynamicky podľa potrieb aplikácie.



Obr. 3.1.4 Ukážka aplikácie s navigačným panelom v hornej časti
Zdroj: vlastné spracovanie

- **Bočné panely (sidebars)** – môžu sa nachádzať naľavo, napravo alebo vnútri karty. Niektoré je možné skryť a používateľ si ich môže zobraziť podľa potreby.
- **Panely (panels)** – môžu byť plávajúce, takže sa dajú umiestniť kdekoľvek na obrazovke a presúvať podľa potreby.

- **Záložky (tabs)** – umožňujú rozdeliť obsah aplikácie do samostatných sekcií.

Shiny ponúka tri hlavné typy rozvrhnutia stránky a to **fluidné**, **fixné** a **vyplniteľné**. Fluidný grid systém je odporúčaný pre väčšinu aplikácií a je predvolený v Shiny funkciách ako `page_navbar()` a `page_sidebar()`. Každý z týchto systémov používa 12-stĺpcový grid na rozloženie prvkov. Líšia sa však v spôsobe, akým interagujú s oknom prehliadača:

- **Fluidný systém** vždy zaberá celú šírku webovej stránky a dynamicky prispôsobuje svoje komponenty podľa veľkosti okna.
- **Fixný systém** má predvolenú širku 940 pixelov, pričom sa môže prispôsobiť na 724 px alebo 1170 px.
- **Vyplniteľný systém** vždy zaberá celú šírku aj výšku stránky a dynamicky mení veľkosť svojich komponentov podľa veľkosti okna.[23]

Používatelia môžu interagovať s webovou stránkou v Shiny aplikácii prostredníctvom vstupných prvkov. Tie môžu vykonávať niekoľko rôznych akcií ako napríklad kliknutie na tlačidlo, zadanie textu, výber z možností, použitie posuvníkov, a i.

Tab. 3.1.1 Prehľad vstupných elementov v Shiny

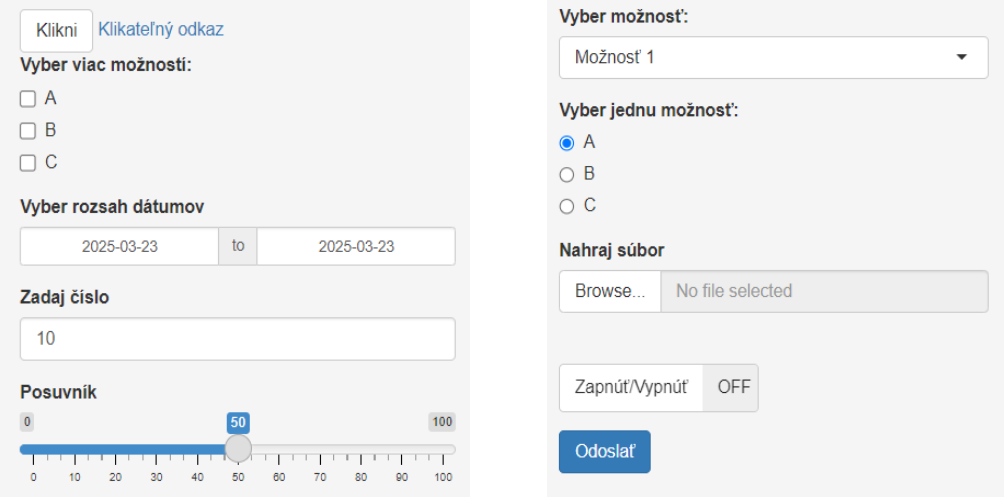
Kategória	Funkcia	Funkcionalita
Tlačidlá a odkazy	`actionButton()`	Spustenie akcie kliknutím
	`actionLink()`	Klikateľný odkaz
Zaškrtávacie polia	`checkboxInput()`	Jedno políčko (áno/nie)
	`checkboxGroupInput()`	Skupina políčok
Dátumové vstupy	`dateInput()`	Výber jedného dátumu
	`dateRangeInput()`	Výber rozsahu dátumov
Číselné vstupy	`numericInput()`	Zadanie čísla
	`sliderInput()`	Posuvník s číselným rozsahom
	`sliderInput(range = TRUE)`	Posuvník na výber rozsahu
Výberové prvky	`selectInput()`	Rozbaľovací zoznam
	`radioButtons()`	Výber jednej možnosti
Textové polia	`textInput()`	Jednoduché textové pole

[23]POSIT. *Arrange Elements – Shiny* [online]. [cit. 25. 03. 2025]. Dostupné z: https://shiny.posit.co/r/layouts/arrange/.

	`passwordInput()`	Pole na heslo
Ostatné	`fileInput()`	Nahrávanie súborov
	`switchInput()`	Prepínač áno/nie
	`submitButton()`	Tlačidlo na odoslanie vstupov

Zdroj: vlastné spracovanie podľa https://shiny.posit.co/r/components/

Na základe vstupných údajov sú potom generované výsledky v rôznych formách, ako napríklad tabuľky, mapy, grafy, a i.

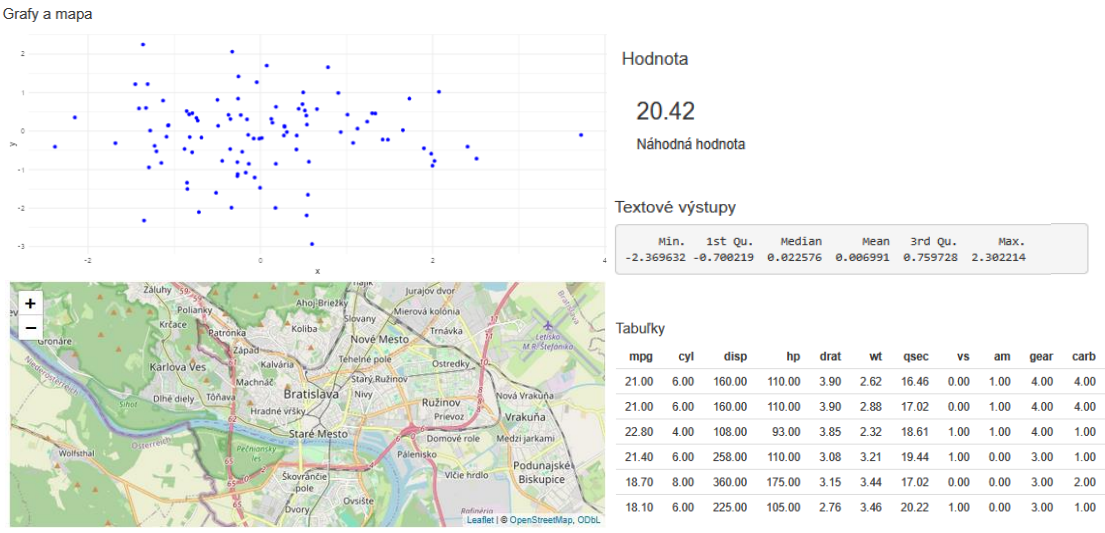


Obr. 3.1.5 Ukážka vstupných elementov v Shiny
Zdroj: vlastné spracovanie

Tab. 3.1.2 Prehľad výstupných elementov v Shiny

Kategória	Funkcia	Funkcionalita
Textové	`textOutput()`	Zobrazí jednoduchý text
	`verbatimTextOutput()`	Zobrazí predformátovaný text
Tabuľkové	`tableOutput()`	Základná tabuľka
	`dataTableOutput()`	Interaktívna tabuľka
Grafické	`plotOutput()`	Klasická graf (`ggplot2`)
	`plotlyOutput()`	Interaktívny graf (`plotly`)
	`leafletOutput()`	Interaktívna mapa
Ostatné	`imageOutput()`	Zobrazenie obrázka
	`valueBoxOutput()`	Informačný box s hodnotou

Zdroj: vlastné spracovanie podľa https://shiny.posit.co/r/components/



Obr. 3.1.6 Ukážka výstupných elementov v Shiny
Zdroj: vlastné spracovanie

3.1.2 Server

Serverová časť v Shiny aplikácii zohráva kľúčovú úlohu, pretože spracováva vstupy od používateľa, vykonáva výpočty a generuje výstupy, ktoré sa zobrazujú v používateľskom rozhraní. Každá Shiny aplikácia má serverovú funkciu, ktorá obsahuje logiku aplikácie a definuje, ako sa majú spracovávať dáta na základe interakcií používateľa.

Na generovanie výstupov sa v Shiny používajú špeciálne `render` funkcie, ktoré určujú typ výstupu. Každý výstup musí mať v UI svoj zodpovedajúci prvok `*Output()`, ktorý ho zobrazí:

- `renderText()` = `textOutput()`.
- `renderTable()` = `tableOutput()`.
- `renderPlot()` = `plotOutput()`.
- `renderLeaflet()` = `leafletOutput()`.

Používateľské vstupy sú v serverovej časti dostupné cez premennú `input`, pričom sa na hodnoty odkazuje pomocou ich ID. Ak má napríklad posuvník v UI identifikátor `slider1`, jeho hodnotu v serverovej časti získame ako `input$slider1`.

3.1.3 Reaktívne prvky

Pri vytváraní interaktívnej aplikácie je potrebné explicitne vyznačiť tie časti kódu, ktoré majú byť reaktívne, aby sa tým zabezpečila ich aktualizácia pri zmene vstupných údajov. Je to z dôvodu, že Shiny automaticky nerozoznáva, ktoré časti kódu sa majú znova vykonať pri zmene vstupov.

Na dosiahnutie reaktivity Shiny používa:

- **Reaktívne hodnoty** – predstavujú dáta, ktoré sa môžu meniť v priebehu času. Majú referenčnú sémantiku, čo znamená, že ak dôjde k ich zmene, všetky časti kódu závislé od týchto hodnôt sa automaticky aktualizujú. Sú definované pomocou dvoch hlavných funkcií:
  - `reactiveVal()` – uchováva jednu konkrétnu hodnotu.
  - `reactiveValues()` – umožňuje ukladať viaceré hodnoty v rámci jednej premennej.
- **Reaktívne výrazy** – umožňujú spracovanie a transformáciu reaktívnych hodnôt. Označujú sa pomocou funkcie `reactive()`. Majú dve dôležité vlastnosti:
  - *Lenivosť (laziness)* – vykonajú sa iba vtedy, keď to je potrebné.
  - *Ukladanie do vyrovnávacej pamäte (caching)* – ak sú zavolané dvakrát za sebou bez zmeny vstupov, namiesto nového výpočtu sa vráti predchádzajúci výsledok.
- **Pozorovateľov (observers)** – reagujú na zmeny v reaktívnom prostredí a vykonávajú vedľajšie efekty, ako napríklad zapisovanie dát do súboru alebo zobrazovanie upozornení. Na rozdiel od reaktívnych výrazov sa vykonajú okamžite ako dôjde k zmene ich vstupných hodnôt. Používajú sa funkcie `observe()` a `observeEvent()`.

Pri práci s reaktivitou môžu nastať situácie, keď chceme zamedziť zbytočnému opakovaniu výpočtov. Na tento účel slúžia funkcie:

- `isolate()` – umožňuje získať hodnotu reaktívnej premennej bez vytvorenia reaktívnej závislosti. To znamená, že výraz nebude spúšťať aktualizáciu pri zmene hodnoty.

- `invalidateLater()` – zabezpečí automatické prepočítanie reaktívneho výrazu po uplynutí určitého časového úseku.[24]

3.1.4 Shinydashboard

Pri návrhu našej aplikácie som použila knižnicu **shinydashboard**, ktorá rozširuje základné možnosti frameworku Shiny o prehľadné používateľské rozhranie v štýle moderných dashboardov. Táto knižnica umožňuje jednoduché vytváranie panelov, navigačného menu, a vizuálne atraktívneho rozloženia obsahu, čím zlepšuje používateľskú skúsenosť a zjednodušuje organizáciu dát.

Ako pri každej inej knižnici, vykonáme inštaláciu pomocou príkazu `install.packages("shinydashboard")`. Potom už ju pomocou príkazu `library (shinydashboard)` môžeme používať. Rozhranie dashboardu je postavené na HTML prvkoch, ktoré Shiny generuje pomocou funkcií `div()` a `p()`. Každý dashboard sa skladá z troch častí:

- **Hlavička (header)** – obsahuje názov a môže obsahovať používateľské menu alebo iné rôzne informačné prvky.
- **Bočný panel (sidebar)** – umožňuje navigáciu v aplikácii a často obsahuje vstupné ovládacie prvky.
- **Hlavná časť (body)** – zobrazujú sa v nej výstupy aplikácie.

```
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody())

server <- function(input,output) { }
shinyApp(ui,server)
```

Obr. 3.1.7 Kód základnej štruktúry dashboardu
Zdroj: vlastné spracovanie

[24] WICKHAM, H. *Mastering Shiny*. Reactive building blocks [online]. 2021 [cit. 25. 03. 2025].

Obsah sa v rámci tela dashboardu usporadúva pomocou Bootstrap grid systému, kde sa celá šírka rozdeľuje na 12 stĺpcov. Rozloženie môže byť:

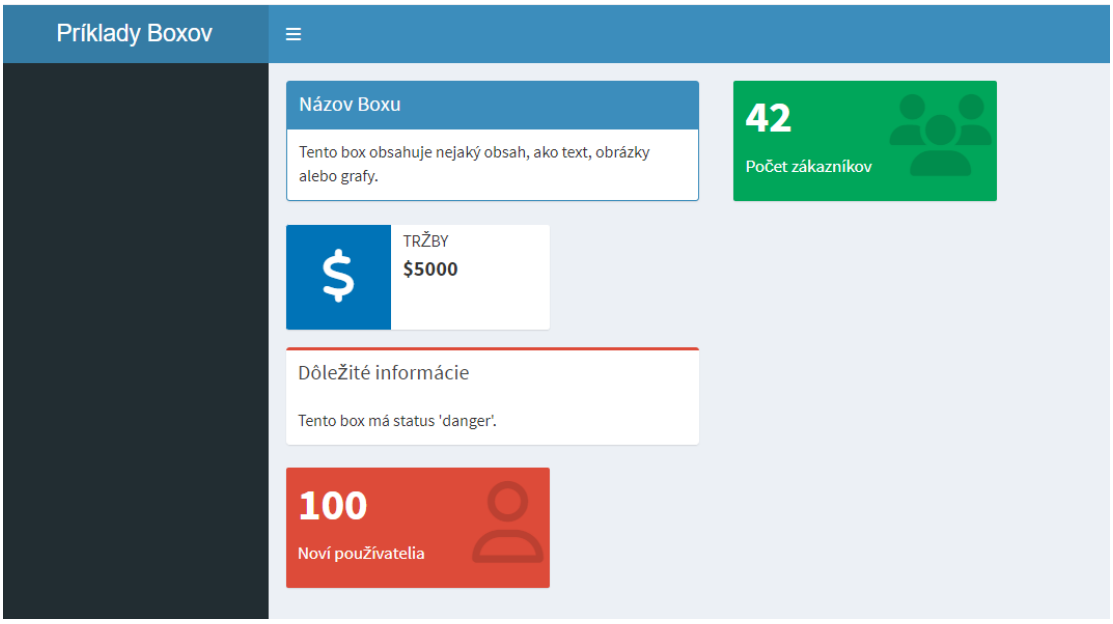
- **Riadené riadkami** (`fluidRow()`) – zaručí zarovnanie horného okraja boxov.
- **Riadené stĺpcami** (`column()`) – umožňuje flexibilnejšie usporiadanie.[25]

V knižnici **shinydashboard** sa obsah organizuje pomocou boxov. Každý box môže obsahovať text, tabuľky, grafy alebo iné prvky a umožňuje nastavenie rôznych vlastností.

Tab. 3.1.3 Typy boxov v shinydashboard

Typ	Popis	Hlavné vlastnosti	Príklad kódu
`box()`	Základný box na organizáciu obsahu. Môže obsahovať text, tabuľky, grafy a ďalšie prvky.	title, width, solidHeader, status	`box(title = "Názov", width = 6, status = "primary", "Obsah")`
`valueBox()`	Zvýrazňuje kľúčové metriky alebo hodnoty v dashboarde.	value, subtitle, icon, color	`valueBox(42, "Počet zákazníkov", icon = icon("users"), color = "green")`
`infoBox()`	Podobný valueBox(), ale s viac informáciami a väčším priestorom na text.	title, value, icon, color	`infoBox("Tržby", "$5000", icon = icon("dollar-sign"), color = "blue")`

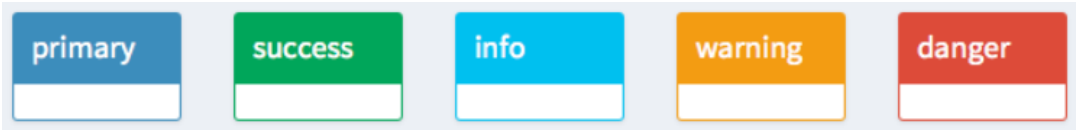
Zdroj: vlastné spracovanie podľa https://rstudio.github.io/shinydashboard/structure.html#boxes



Obr. 3.1.8 Vizualizácia boxov v shinydashboard
Zdroj: vlastné spracovanie

[25] RStudio. *Shiny dashboard: Structure* [online]. GitHub, 2021 [cit. 25. 03. 2025]. Dostupné z: https://rstudio.github.io/shinydashboard/structure.html

Vzhľad boxov môžeme upravovať pomocou farieb a statusov. Farby sa používajú pre boxy ako `valueBox()` a `infoBox()` na zvýraznenie hodnoty, zatiaľ čo statusy sa využívajú pre obyčajné `box()` na určenie vizuálnej hierarchie.[26]



Obr. 3.1.9 Statusy boxov v shinydashboard,
Zdroj:https://rstudio.github.io/shinydashboard/images/statuses.png

3.2 Charakteristika úloh Management Science

Väčšina optimalizačných úloh lineárneho programovania sa rieši **pomocou simplexovej metódy**, ktorú v roku 1947 vyvinul George B. Dantzig. Jej cieľom je nájsť optimálne hodnoty nezáporných premenných, ktoré vyhovujú sústave lineárnych rovníc a minimalizujú alebo maximalizujú hodnotu cieľovej funkcie. Proces riešenia zahŕňa iteratívne zlepšovanie základného prípustného riešenia prostredníctvom prechodu medzi vrcholmi mnohostenu definovaného obmedzeniami problému, až kým sa nedosiahne optimálne riešenie.

Simplexová metóda sa často realizuje v dvoch fázach:

- **Fáza I**: Cieľom je nájsť počiatočné bázické prípustné riešenie. Ak takéto riešenie nie je zrejmé z pôvodnej sústavy rovníc, zavádzajú sa umelé premenné, ktoré umožnia transformovať sústavu na kanonický tvar. Následne sa pomocou simplexového algoritmu hľadá prípustné riešenie minimalizáciou súčtu týchto umelých premenných. Ak sa podarí nájsť takéto riešenie a všetky umelé premenné sú nulové, prechádza sa k Fáze II.
- **Fáza II**: V tejto fáze sa zameriavame na optimalizáciu účelovej funkcie, pričom vychádzame z prípustného riešenia nájdeného v Fáze I. Iteratívne sa mení bázická

[26] RStudio. *Shiny dashboard: Structure* [online]. [cit. 25. 03. 2025].

množina premenných tak, aby sa postupne zlepšovala hodnota účelovej funkcie, až kým sa dosiahne optimálne riešenie.[27]

Významné vlastnosti **simplexovej metódy** zahŕňajú:

- **Univerzálnosť**: Je vhodná na riešenie širokej škály úloh lineárneho programovania bez potreby predchádzajúcich znalostí o štruktúre sústavy.
- **Flexibilita**: Umožňuje zavedenie umelých premenných na získanie východiskového prípustného riešenia, čo je užitočné pri práci so sústavami, ktoré nie sú v kanonickom tvare.
- **Efektívnosť**: Aj keď v najhoršom prípade môže mať exponenciálnu časovú zložitosť, v praxi často poskytuje riešenia v rozumnom čase.

3.2.1 Výrobný problém

Simplexová metóda je častý nástroj na riešenie úloh lineárneho programovania, ktorý možno aplikovať na širokú škálu problémov z praxe. Reálne situácie však vyžadujú komplexné modely obsahujúce stovky rozhodovacích premenných a obmedzení, preto sa pri vysvetľovaní tejto metódy používajú iba ilustračné prípady, ktorých cieľ je iba načrtnúť možnosť ich riešenia.[28]

Medzi typické aplikácie patrí **výrobný problém**, ktorý rieši alokáciu exogénnych zdrojov (kapacity strojov, pracovná sila, materiál) s cieľom maximalizovať zisk alebo minimalizovať náklady. Formulácia takéhoto prípadu by mohla byť nasledovná:

Firma vyrába sviečky v tvare srdca a kruhu, pričom má k dispozícii 210 pracovných hodín, 180 kg vosku a 150 strojového času. Na výrobu srdca sa spotrebujú 4 kg vosku, 2 pracovné hodiny a 3 h strojového času. Na výrobu kruhu sa využijú 2 kg vosku, 3 pracovné hodiny a 1 strojová hodina. Zisk zo srdiečkovej sviečky je 40 € a kruhovej 30 €. Cieľom firmy je určiť optimálny výrobný plán maximalizujúci zisk.

Ako prvý krok si túto úlohu prepíšeme do matematického modelu nasledovne:

[27] DANTZIG, George Bernard. *Lineárne programovanie a jeho rozvoj*. Vydanie prvé. Bratislava: Slovenské vydavateľstvo technickej literatúry, 1966, s. 120–133.
[28] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 70.

Maximalizujeme: $z(x) = 40x_1 + 30x_2$

Za podmienok: $2x_1 + 3x_2 \leq 210$

$4x_1 + 2x_2 \leq 180$

$3x_1 + x_2 \leq 150$

$x_1, x_2 \geq 0$

Následne si vytvoríme začiatočnú **simplexovú** tabuľku so zavedením umelých premenných s_1, s_2, s_3, ktoré reprezentujú nevyužité zdroje. Cieľová funkcia sa prepíše do tvaru, kde všetky premenné sú na jednej strane.

Tab. 3.2.1 Začiatočná simplexová tabuľka výrobného problému

Základ	x_1	x_2	s_1	s_2	s_3	**RHS**
s_1	2	3	1	0	0	210
s_2	4	2	0	1	0	180
s_3	3	1	0	0	1	150
Z	-40	-30	0	0	0	0

Zdroj: vlastné spracovanie

V riadku cieľovej funkcie vyberieme najzápornejší koeficient, ktorý vstúpi do bázy, keďže jeho zvýšenie zlepší hodnotu *Z*. Pre každý riadok obmedzení sa vypočíta pomer RHS/príslušný koeficient vo vedúcom riadku, pričom riadok s najmenším pomerom určuje premennú, ktorá opustí bázu. Vybraný vedúci riadok sa normalizuje na 1 (celý riadok sa vydelí vedúcim prvkom) a ostatné riadky sa upravia tak, aby vo vedúcom stĺpci boli nuly. Ak sú všetky koeficienty v riadku *Z* nezáporné, tak sme našli optimálne riešenie. Ak sú koeficienty záporné, tak sa vstupuje do ďalšej iterácie **simplexového algoritmu**.[29]

Optimálna tabuľka nášho príkladu vyzerá takto:

Tab. 3.2.2 Optimálna tabuľka výrobného problému

Základ	x_1	x_2	s_1	s_2	s_3	**RHS**
x_2	0	1	0,5	-0,25	0	60
x_1	1	0	-0,25	0,375	0	15
s_3	0	0	0,25	-0,875	1	45
Z	0	0	5	7,5	0	2400

Zdroj: vlastné spracovanie

[29] DANTZIG, G. *Lineárne programovanie a jeho rozvoj*. 1966, s. 133-136.

Optimálny výrobný plán pre podnik je výroba sviečok v tvare srdca v počte 15 ks a v tvare kruhu v počte 60 ks. Takýto plán vygeneruje pre firmu zisk 2 400 €.

3.2.2 Dopravná úloha

Dopravné úlohy predstavujú optimalizačný problém, ktorého cieľom je minimalizovať náklady na prepravu pri súčasnom splnení požiadaviek odberateľov a rešpektovaní kapacitných možností dodávateľov. Tento problém možno formulovať ako úlohu lineárneho programovania. Optimalizácia sa týka riadenia tokov medzi jednotlivými výrobnými subjektmi, ako aj medzi dodávateľmi a odberateľmi, pričom hlavným cieľom je dosiahnuť efektívne rozdelenie zdrojov pri čo najnižších nákladoch.

Dopravná úloha spočíva v distribúcii jedného materiálu z *m* zdrojov k *n* odberateľom. Každý dodávateľ *i* je schopný dodať b^d_i (kde $i = 1,2, ...m$), zatiaľ čo každý odberateľ *j* môže prijať maximálne množstvo b^o_j (kde $j = 1,2, ...n$). Prepravné náklady na jednotku materiálu medzi *i*-tým dodávateľom a *j*-tym odberateľom bude označené c_{ij}. Cieľom je nájsť taký spôsob distribúcie materiálu, aby náklady na prepravu boli minimálne.[30]

Poznáme dva typy dopravných úloh:

- **Vybilancovaná** – platí, že celková kapacita dodávateľov sa presne rovná dopytu odberateľov, t.j. $\sum_{i=1}^{m} b_i^d = \sum_{j=1}^{n} b_j^o$. Formulácia príkladu, na ktorom si ukážeme tento typ úlohy: Podnik má 2 pobočky v rôznych mestách pričom odoberá tovar v celkovom množstve 500 ks od dodávateľa, ktorý ho odosiela z 3 rôznych skladov. *Pobočka A* odoberá 350 ks a *pobočka B* 150 ks. Skladové kapacity dodávateľa sú: *sklad1* = 140 ks, *sklad2* = 170 ks, *sklad3* = 190 ks. Prepravné náklady sú zapísane v matici $\boldsymbol{C}$:

$$\boldsymbol{C} = \begin{bmatrix} 7 & 6 \\ 2 & 3 \\ 8 & 4 \end{bmatrix}$$

Zápis tejto úlohy ako ÚLP:

$$\min z(x) = 7x_{11} + 6x_{12} + 2x_{21} + 3x_{22} + 8x_{31} + 4x_{32}$$

[30] BREZINA, I. a PEKÁR, J. *Management Science.* 2024, s. 144.

$$x_{11} + x_{12} = 140$$

$$x_{21} + x_{22} = 170$$

$$x_{31} + x_{32} = 190$$

$$x_{11} + x_{21} + x_{31} = 350$$

$$x_{12} + x_{22} + x_{32} = 150$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0$$

- **Nevybilancovaná** – platí, že celková kapacita dodávateľov sa nerovná dopytu odberateľov, t.j. $\sum_{i=1}^{m} b_i^d \neq \sum_{j=1}^{n} b_j^o$. Formuláciu predošlej úlohy upravíme tak, že kapacity skladov budú: *sklad1* = 250 ks, *sklad2* = 300 ks, *sklad3* = 190 ks, pričom ostatné údaje ostanú nezmenené. Teraz sa stáva úloha nevybilancovanou, pretože celková kapacita skladov sa nerovná celkovej kapacite odberateľov, v resp. $\sum_{i=1}^{m} b_i^d > \sum_{j=1}^{n} b_j^o$ (740>500). Zápis ÚLP sa preto mení na:

$$\min z(x) = 7x_{11} + 6x_{12} + 2x_{21} + 3x_{22} + 8x_{31} + 4x_{32}$$

$$x_{11} + x_{12} \leq 250$$

$$x_{21} + x_{22} \leq 300$$

$$x_{31} + x_{32} \leq 190$$

$$x_{11} + x_{21} + x_{31} = 350$$

$$x_{12} + x_{22} + x_{32} = 150$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0$$

3.2.3 Hľadanie najkratšej cesty

Hľadanie najkratšej cesty je jedným zo základných problémov v teórii grafov, kde cieľom je nájsť trasu s minimálnymi nákladmi medzi dvoma bodmi v sieti. Tento problém sa modeluje ako ohodnotený graf, kde uzly predstavujú body (napr. mestá) a hrany reprezentujú spojenia medzi nimi s priradenými nezápornými hodnotami vyjadrujúcimi náklady na presun.

Najčastejšie používaným algoritmom na riešenie tohto problému je **Dijkstrov algoritmus**, ktorý efektívne vyhľadáva najkratšiu cestu z jedného východiskového bodu do všetkých ostatných vrcholov v grafe.

Dijkstrov algoritmus funguje na princípe postupného spracovania vrcholov:

1. Inicializácia – na začiatku sa nastaví dĺžka cesty z východiskového vrcholu do seba na 0, zatiaľ čo dĺžky ciest do ostatných vrcholov sú považované za nekonečné.
2. Spracovanie vrcholov – algoritmus vyberie vrchol s aktuálne najkratšou známou cestou a označí ho ako „spracovaný“.
3. Aktualizácia susedov – pre každý nespracovaný susedný vrchol sa prehodnotí jeho vzdialenosť cez práve spracovaný vrchol. Ak je nová cesta kratšia ako doteraz známa, hodnota sa aktualizuje.
4. Opakovanie krokov 2-3 – tento proces pokračuje, kým nebudú spracované všetky vrcholy alebo nebude nájdená najkratšia cesta do cieľového vrcholu.

Po ukončení algoritmu sú známe najkratšie cesty z východiskového bodu do všetkých dosiahnuteľných vrcholov. Hlavnou nevýhodou **Dijkstrovho algoritmu** je, že nefunguje správne v grafoch so zápornými hranami, preto je pre taký prípad potrebné použiť iné metódy, ako napríklad Bellman-Fordov algoritmus.[31]

Použijeme **Dijkstrov algoritmus** na orientovanom ohodnotenom grafe z obr. 3.2.1, v ktorom budeme hľadať najkratšiu cestu z vrcholu u_1 do všetkých ostatných vrcholov.

1. Vyberieme začiatočný bod u_1 a inicializujeme vzdialenosti do ostatných vrcholov:
   - $u_1 = 0$.
   - $u_2 = 4$.
   - $u_3 = 2$ – najmenšia vzdialenosť, budeme spracovávať ako prvý.
   - $u_4 = \infty$ (neexistuje žiadna priama cesta).
   - $u_5 = \infty$.
2. Spracujeme vrchol u_3:
   - Z u_3 vedie cesta do $u_2 = 3$.

[31] KSP. *Dijkstrov algoritmus* [online]. 2020 [cit. 29. 03. 2025]. Dostupné z: https://www.ksp.sk/kucharka/dijkstra/

- Skontrolujeme, či je výhodnejšie použiť túto cestu namiesto pôvodnej hodnoty = min(4, 2+3) = 4 . Vzdialenosť ostáva nezmenená a je momentálne najmenšia, preto vrchol u2 spracujeme ako ďalší.
- Z u_3 vedie cesta do u_5 = 6, preto aktualizujeme vzdialenosť 2+6 = 8.

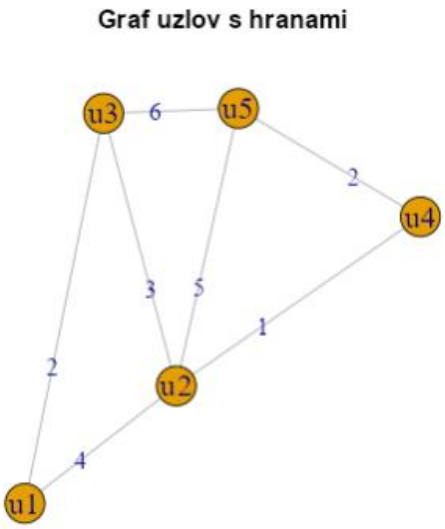
3. Spracujeme vrchol u_2:
   - Z u_2 vedie cesta do u_4 = 1.
   - Opäť skontrolujeme, či sa oplatí použiť túto cestu = min(∞,4+1) = 5, a túto vzdialenosť aktualizujeme, zároveň je najmenšia, takže ako ďalší spracujeme u_4.
   - Z u_2 vedie cesta do u_5 = 5.
   - Kontrola min(8,4+5) = 8, vzdialenosť ostáva nezmenená.
4. Spracujeme vrchol u_4:
   - Z u_4 vedie cesta do u_5 = 2.
   - Kontrola min(8,5+2) = 7, vzdialenosť aktualizujeme a spracujeme posledný vrchol.
5. Spracujeme vrchol u_5:
   - Žiadne nové skrátenie ciest.

Zistili sme, že najkratšie vzdialenosti z u_1 do všetkých ostatných sú:

Tab. 3.2.3 Najkratšie vzdialenosti z u_1 Dijkstrov algoritmus

Vrchol	Najkratšia vzdialenosť z u_1
u_1	0
u_2	4
u_3	2
u_4	5
u_5	7

Zdroj: vlastné spracovanie



Obr. 3.2.1 Orientovaný ohodnotený graf pre príklad Dijkstrovho algoritmu
Zdroj: vlastné spracovanie

3.2.4 Hľadanie najkratšej okružnej cesty

Hľadanie najkratšej okružnej cesty, známe aj ako **problém obchodného cestujúceho (Traveling Salesman Problem – TSP)**, je klasickou úlohou v teórii grafov. Cieľom je nájsť najkratšiu možnú cestu, ktorá prechádza každým z daných uzlov práve raz a vráti sa do východiskového bodu.

Metódy riešenia problému obchodného cestujúceho:

- **Optimalizačné – Littleho metóda**: Ide o exaktný algoritmus založený na vetvení a ohraničovaní (branch and bound), ktorý systematicky skúma všetky možné permutácie ciest a postupne vylučuje tie, ktoré nemôžu viesť k optimálnemu riešeniu. Táto metóda je výpočtovo náročná a vhodná najmä pre problémy s menším počtom uzlov.
- **Heuristické – Algoritmus najbližšieho suseda (Nearest Neighbor Algorithm – NNA)** : Tento algoritmus začína v ľubovoľnom uzle a v každom kroku sa presunie do najbližšieho ešte nenavštíveného uzla. Tento postup sa opakuje kým nie sú navštívené všetky uzly, a následne sa vráti do východiskového bodu. Hoci je tento algoritmus jednoduchý a rýchly, neposkytuje vždy optimálne riešenie.

Postup riešenia úlohy s použitím algoritmu najbližšieho suseda:

1. Ľubovoľne alebo podľa určitého kritéria sa vyberie začiatočný uzol.
2. Z aktuálneho uzla sa vyberie ten, ktorý je najbližší a ešte nebol navštívený.
3. Po presune do vybraného uzla sa tento označí ako navštívený a stane sa novým východiskovým uzlom pre ďalší krok.
4. Postup sa opakuje, kým nie sú navštívené všetky uzly.
5. Po navštívení všetkých miest sa trasa uzavrie návratom do počiatočného uzla.

Aplikujeme tento postup na príklad s 5 miestami a týmito vzdialenosťami:

Tab. 3.2.4 Tabuľka vzdialeností uzlov príklad algoritmu najbližšieho suseda

	u_1	u_2	u_3	u_4	u_5
u_1	0	10	15	20	25
u_2	10	0	35	25	30
u_3	15	35	0	30	20
u_4	20	25	30	0	15
u_5	25	30	20	15	0

Zdroj: vlastné spracovanie

1. Štart z uzla u_1.
2. Najbližší sused u_1 je u_2 = vzdialenosť 10. Cesta $u_1 \rightarrow u_2$.
3. Najbližší sused u_2 je u_4 = vzdialenosť 25. Cesta $u_1 \rightarrow u_2 \rightarrow u_4$.
4. Najbližší sused u_4 je u_5 = vzdialenosť 15. Cesta $u_1 \rightarrow u_2 \rightarrow u_4 \rightarrow u_5$.
5. Najbližší sused u_5 je u_3 = vzdialenosť 20. Cesta $u_1 \rightarrow u_2 \rightarrow u_4 \rightarrow u_5 \rightarrow u_3$.
6. Návrat z u_3 do u_1 = vzdialenosť 15. Cesta $u_1 \rightarrow u_2 \rightarrow u_4 \rightarrow u_5 \rightarrow u_3 \rightarrow u_1$.

Celková prejdená vzdialenosť je 85 (10+25+15+20+15).

3.3 Tvorba aplikácie Shiny

Pri tvorbe navrhnutej Shiny aplikácie sme použili niekoľko knižníc, ktoré zabezpečujú jej správne fungovanie a zabezpečenie jednotlivých funkcionalít:

- `shiny` – základný framework pre vytvorenie aplikácie v R. Umožňuje prepojenie medzi užívateľským rozhraním a serverovou logikou, čím zabezpečuje dynamické spracovanie vstupov a vizualizáciu výstupov.
- `shinydashboard` – aplikáciu obohatil o prehľadnú dashboardovú štruktúru, pre zvýšenie užívateľského zážitku.
- `lpSolve` – balík určený na riešenie úloh lineárneho programovania. V aplikácii bol použitý pri implementácii **výrobného problému** a **dopravnej úlohy**.
- `TSP` – knižnica špecializovaná na riešenie **problému obchodného cestujúceho**. Obsahuje viaceré heuristické metódy, z ktorých sme využili metódu najbližšieho suseda na nájdenie optimálnej okružnej cesty medzi uzlami v grafe.
- `igraph` – rozsiahla knižnica pre prácu s grafmi a sieťami. V aplikácii bola použitá pri implementácii **Dijkstrovho algoritmu** na **hľadanie najkratšej cesty** v orientovanom grafe.
- `leaflet` – knižnica pre tvorbu interaktívnych máp v R, použitá na vizualizáciu riešenia dopravnej úlohy.
- `ggplot2` – všestranne využiteľný balík na vizualizáciu dát v R, využitý v grafoch riešenia optimalizačných úloh.
- `DT` – knižnica pre tvorbu interaktívnych tabuliek. Jej hlavnou výhodou je možnosť priamej editácie tabuliek v aplikácii, čo bolo nevyhnutné pri zadávaní matíc vzdialeností v úlohách hľadania najkratších ciest.

3.3.1 Výrobný problém

Výrobný problém je v aplikácii implementovaný ako optimalizačný model, ktorý maximalizuje zisk alebo minimalizuje náklady pri výrobe rôznych produktov, pričom zohľadňuje dostupné suroviny, výrobné kapacity, časové obmedzia a možné závislosti medzi produktmi.

V `ui` časti je rozhranie definované cez `tabItem(tabName = "vyroba")`, kde sa nachádzajú vstupné prvky pre nastavenie parametrov modelu:

- *Základné nastavenia* – interaktívne posuvníky pre výber počtu produktov a surovín.
- *Zisk* – užívateľ definuje zisk z každého produktu (`product_profit`).
- *Náklady* – možnosť zahrnúť výrobné náklady (`include_costs`) a zadať ich hodnoty (`production_cost`).
- *Materiálové kapacity a spotreba materiálu* – možnosť zadať dostupné množstvo surovín (`material_availability`) a definovať spotreby surovín pre jednotlivé produkty (`material_usage`).
- *Závislosti medzi produktmi* – možnosť definovať vzájomné závislosti medzi produktmi (`dependency_checkbox`) a určiť množstvo jedného produktu potrebné na výrobu iného produktu (`product_dependecies`).
- *Optimalizačný panel* – užívateľ vyberá medzi maximalizáciou zisku alebo minimalizáciou nákladov (`objective`), tlačidlo spustenia optimalizácie (`optimize_v`).

V časti serverovej logiky sa tieto užívateľské vstupy získavajú dynamicky cez `renderUI()`, pričom sa vytvárajú početné vstupné polia pre zisk, náklady, suroviny a závislosti medzi produktmi. Každý parameter je generovaný pomocou `lapply()`, čím sa automaticky prispôsobuje zadanému počtu produktov a surovín.

Po stlačení tlačidla „optimalizovať“ sa zostaví model nasledovne:

1. Definuje sa cieľová funkcia (maximalizácia zisku/minimalizácia nákladov).
2. Vytvoria sa obmedzenia – každý produkt nesmie prekročiť dostupné množstvo suroviny, časové obmedzenie, alebo musí byť zabezpečená dostatočná produkcia produktu potrebného na iný produkt.
3. Riešenie modelu pomocou `lpSolve`:

```
lp_result <- lp(
          direction = "max",
          objective.in = objective_function,
          const.mat = all_constraints,
          const.dir = all_directions,
          const.rhs = all_rhs,
          all.int = TRUE
        )
```

Obr. 3.3.1 Riešenie pomocou lpSolve
Zdroj: vlastné spracovanie

- `objective_function` – funkcia, ktorú optimalizujeme.
- `constraints` – matica obmedzení.
- `directions` - smer nerovností (<= pre obmedzenia surovín, >= pre závislosti).
- `rhs` – pravá strana nerovností (dostupné množstvo surovín, časové kapacity).
- `all.int = TRUE` – riešenie musí byť v celých číslach (nemôžeme vyrobiť 2,5 produktu).

4. Vizualizácia výsledkov – tabuľka a stĺpcový graf zobrazujúci optimálnu výrobu jednotlivých produktov.

3.3.2 Dopravná úloha

V implementácii **dopravnej úlohy** si môže používateľ interaktívne nastaviť počet dodávateľov a zákazníkov, špecifikovať kapacity, dopyty a nákladové či časové matice, a na základe optimalizačného cieľa vypočítať optimálne rozdelenie prepravy.

Rozhranie `ui` definované cez `tabItem(tabName = "transport")`, a pozostáva z nasledujúcich častí:

- *Základné nastavenia* – interaktívne posuvníky pre výber počtu dodávateľov a zákazníkov.
- *Kapacity dodávateľov* – numerické vstupy pre maximálnu kapacitu každého dodávateľa (`supplier_capacity`).
- *Dopyty zákazníkov* – pole pre zadanie požadovaného množstva pre každého zákazníka (`customer_demand`).
- *Nákladová matica* – matice jednotkových nákladov medzi všetkými dodávateľmi a zákazníkmi (`cost_matrix`).
- *Časová matica* – matice časov prepravy medzi všetkými dvojicami (`time_matrix`).
- *Rozšírené možnosti* – obmedzenie maximálneho času doručenia (`max_time`), obmedzenie kapacity vozidiel (`vehicle_capacity`).

- *Optimalizačný panel* – výber optimalizačného scenára z minimalizácie nákladov/času alebo ich kombinácie (`optimize_t`).

Podobne ako pri výrobnom probléme, aj v dopravnej úlohe sa v serverovej časti dynamicky generujú všetky vstupné parametre pomocou `renderUI()` a `lapply()`. Tieto funkcie umožňujú flexibilné vytváranie polí pre každého dodávateľa či zákazníka a komplexných matíc nákladov a časov.

Po kliknutí na „optimalizovať“ aplikácia vykoná:

1. Načíta vstupné hodnoty – kapacity dodávateľov, dopyty zákazníkov, nákladovú a časovú maticu.
2. Aplikuje obmedzenia – ak je aktivované obmedzenie času, vyradí sa preprava, ktorá by trvala príliš dlho.
3. Zvolí optimalizačný cieľ – podľa užívateľského výberu vyberie optimalizačnú maticu.
4. Overí splniteľnosť problému – ak je celková kapacita menšia ako dopyt, zobrazi sa chybové hlásenie.
5. Rieši dopravný problém pomocou funkcie `lp.transport()` z balíka `lpSolve`:

```
result <- lp.transport(
      cost.mat = optimization_matrix,
      direction = "min",
      row.signs = rep("<=", num_suppliers),
      row.rhs = supplier_capacity,
      col.signs = rep("=", num_customers),
      col.rhs = customer_demand,
      integers = NULL
    )
```

Obr. 3.3.2 Riešenie dopravnej úlohy pomocou lp.transport()
Zdroj: vlastné spracovanie

- `cost.mat` – matica optimalizačných kriterií.
- `direction` – špecifikuje optimalizačný cieľ.
- `row.signs` – logika obmedzení <= aby kapacita dodávateľa nemohla byť prekročená.
- `row.rhs` – vektor kapacít každého dodávateľa.
- `col.signs` – dopytové obmedzenia = vyžaduje presné splnenie dopytu.

- `col.rhs` – vektor dopytov každého zákazníka.
- `integers = NULL` – dovolené reálne hodnoty.

6. Vyhodnotí a vizualizuje výsledky – zobrazí tabuľku riešenia, tepelnú mapu (čím silnejšia farba, tým väčšie množstvo), a približné znázornenie prepravných trás (čím hrubšia čiara, tým väčší objem tovaru).

3.3.3 Nájdenie najkratšej a najkratšej okružnej cesty

V poslednej časti aplikácie si môže používateľ interaktívne vytvoriť graf zadaním počtu uzlov a doplnením vzdialeností medzi nimi. Následne je možné:

- **Vypočítať najkratšiu cestu** medzi dvoma uzlami pomocou **Dijkstrovho algoritmu**, ktorý nájde optimálnu trasu na základe minimálnych váh hrán.
- **Vypočítať najkratšiu okružnú cestu (TSP)**, ktorá pokrýva všetky uzly a vracia sa do východiskového bodu, pričom sa snaží minimalizovať celkovú vzdialenosť.

Užívateľské rozhranie pozostáva z dvoch hlavných panelov:

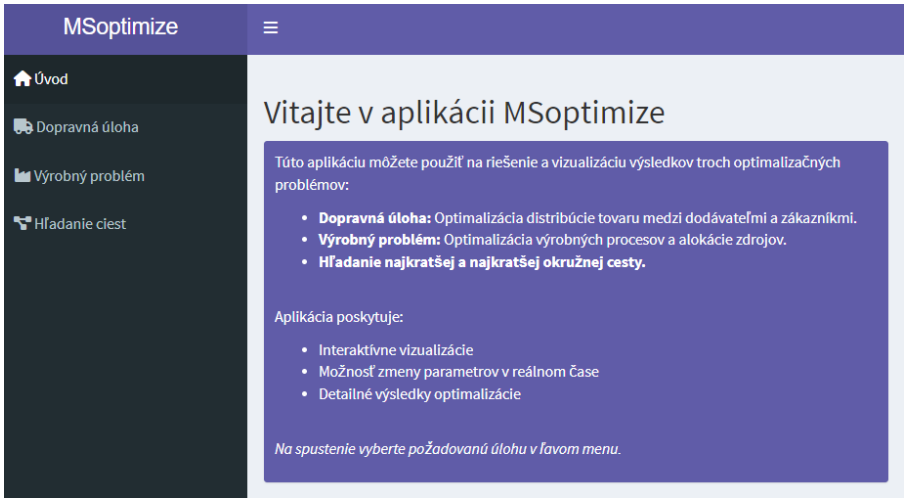
- **Vstupný panel** – pozostáva z nastavenia počtu uzlov, interaktívnej tabuľky pre zadanie vzdialeností, a nástroje pre výpočty.
- **Výstupný panel** – grafická vizualizácia siete a textové výsledky výpočtov.

Postup priebehu serverovej logiky:

1. Inicializácia reaktívnych hodnôt, ktoré uchovávajú stav uzlov a matice vzdialeností a automaticky aktualizujú závislé výpočty.
2. Generovanie grafu a inicializácia vzdialeností s NA hodnotami.
3. Editácia vzdialeností a interaktívna úprava hrán.
4. Vytvorenia grafu pomocou `igraph`.
5. Podľa rozhodnutia používateľa vykoná **dijkstrov algoritmus** pre **nájdenie najkratšej cesty** medzi dvoma uzlami alebo **metódu najbližšieho suseda** pre **problém obchodného cestujúceho**.

4 VÝSLEDKY PRÁCE

Aplikácia dostala názov **MSoptimize**, keďže je zameraná na riešenie optimalizačných úloh v oblasti Management Science. Po spustení sa zobrazí úvodná stránka so základnými informáciami a navigáciou, ktorá používateľa usmerní k výberu požadovanej úlohy cez bočný panel. Po kliknutí na konkrétnu sekciu sa otvorí príslušné rozhranie, kde môže používateľ zadať vstupné parametre a sledovať výsledky.



Obr. 3.3.1 Úvodná stránka aplikácie
Zdroj: vlastné spracovanie

4.1 Overenie funkcionality vytvorenej aplikácie

V tejto časti si ukážeme funkcionalitu vytvorenej aplikácie na konkrétnych príkladoch, pričom správnosť výsledkov overíme porovnaním s manuálnymi výpočtami uvedenými v predchádzajúcich kapitolách práce alebo riešením v prostredí jazyka R.

4.1.1 Aplikácia: Dopravná úloha

Nasledujúci príklad opisuje situáciu, v ktorej logistická firma zabezpečuje prepravu tovaru medzi 4 skladmi (S_1, S_2, S_3, S_4) a 5 odberateľskými miestami (Z_1, Z_2, Z_3, Z_4, Z_5). Sklady a ich kapacity sú:

- S_1 = 620 jednotiek.
- S_2 = 400 jednotiek.
- S_3 = 350 jednotiek.
- S_4 = 545 jednotiek.

Zatiaľ čo odberatelia vyžadujú:

- Z_1 = 235 jednotiek.
- Z_2 = 300 jednotiek.
- Z_3 = 180 jednotiek.
- Z_4 = 420 jednotiek.
- Z_5 = 410 jednotiek.

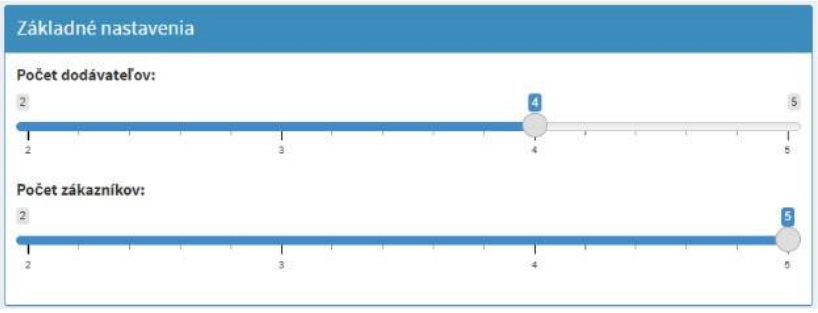
Náklady na prepravu sú zapísané v tab. 4.1.1. Cieľom je optimalizovať rozloženie prepravy tak, aby boli splnené dopyty zákazníkov a kapacity skladov, pričom sa minimalizujú celkové prepravné náklady.

Tab. 4.1.1 Matica nákladov na prepravu jednej jednotky tovaru

7	4	2	8	10
5	3	9	9	4
12	5	7	4	2
8	8	6	11	3

Zdroj: vlastné spracovanie

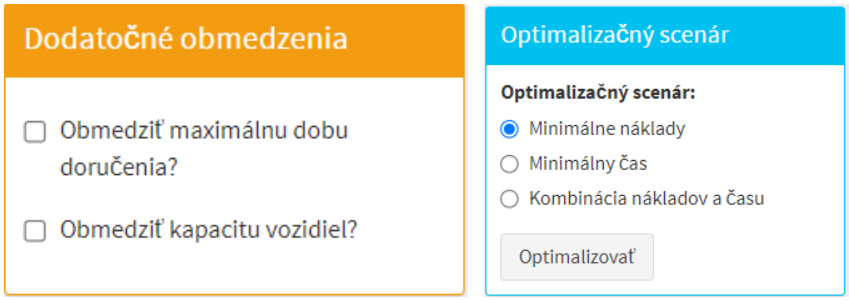
Z bočného panela vyberieme dopravnú úlohu a postupne vkladáme vstupné údaje z definovaného príkladu.



Obr. 4.1.1 Zadávanie vstupných údajov dopravná úloha
Zdroj: vlastné spracovanie

Obr. 4.1.2 Zadávanie vstupných údajov a nákladovej matice dopravná úloha
Zdroj: vlastné spracovanie

Po dokončení vkladania všetkých dostupných informácií si v boxe optimalizačného scenára vyberieme vhodnú možnosť – minimálne náklady.



Obr. 4.1.3 Výber optimalizácie dopravná úloha
Zdroj: vlastné spracovanie

Následne sa nám vo výsledkovom boxe zobrazí tabuľka s riešením, interpretácia, a vizualizácia. Podľa aplikácie predstavujú minimálne náklady pri optimálnom výrobnom pláne sumu 5 760 €. Výrobný plán je znázornený pomocou tepelnej mapy, kde platí, že čím väčšie prepravené množstvo, tým tmavšia farba. K dispozícii je aj približné zobrazenie prepravných trás – každá trasa má inú farbu hrúbka čiary zodpovedá prepravenému množstvu.

Príklad je následne zadaný aj do jazyka R za účelom overenia správnosti výpočtu. Po porovnaní zisťujeme, že výsledky sa zhodujú, čo potvrdzuje správnosť a spoľahlivosť implementovanej optimalizácie.

Výsledky

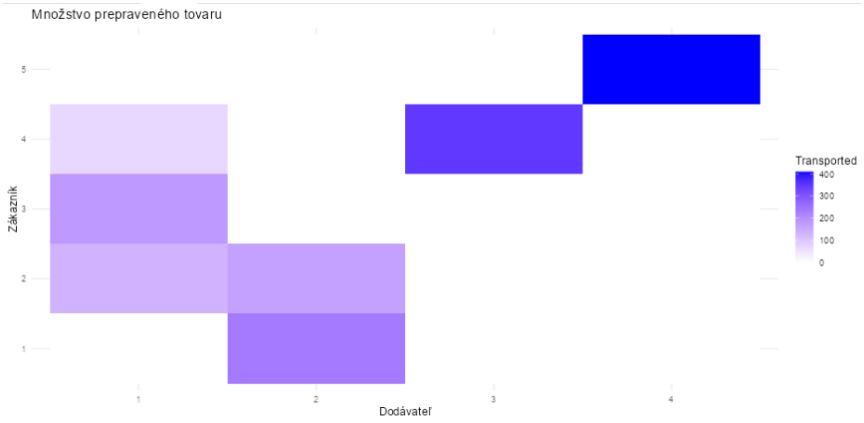
Tabuľka riešenia | Graf | Približné znázornenie prepravných trás

	Zákazník 1	Zákazník 2	Zákazník 3	Zákazník 4	Zákazník 5
Dodávateľ 1	0.00	135.00	180.00	70.00	0.00
Dodávateľ 2	235.00	165.00	0.00	0.00	0.00
Dodávateľ 3	0.00	0.00	0.00	350.00	0.00
Dodávateľ 4	0.00	0.00	0.00	0.00	410.00

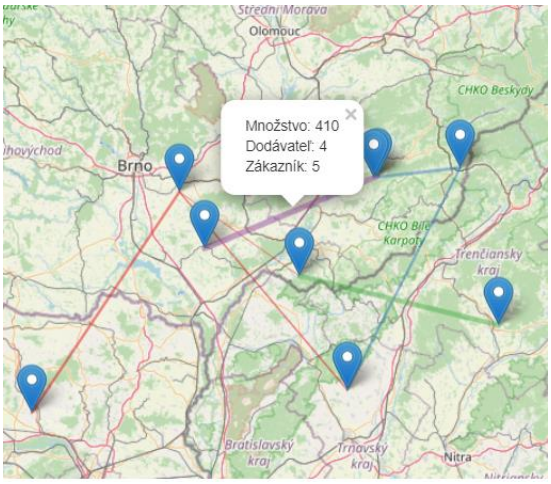
Interpretácia výsledkov

Optimálny prepravný plán: Dodávateľ 1 bude dodávať 135 jednotiek tovaru zákazníkovi 2. Náklady na jednotku sú 4 € a čas dodania je 1 hodín. Dodávateľ 1 bude dodávať 180 jednotiek tovaru zákazníkovi 3. Náklady na jednotku sú 2 € a čas dodania je 1 hodín. Dodávateľ 1 bude dodávať 70 jednotiek tovaru zákazníkovi 4. Náklady na jednotku sú 8 € a čas dodania je 1 hodín. Dodávateľ 2 bude dodávať 235 jednotiek tovaru zákazníkovi 1. Náklady na jednotku sú 5 € a čas dodania je 1 hodín. Dodávateľ 2 bude dodávať 165 jednotiek tovaru zákazníkovi 2. Náklady na jednotku sú 3 € a čas dodania je 1 hodín. Dodávateľ 3 bude dodávať 350 jednotiek tovaru zákazníkovi 4. Náklady na jednotku sú 4 € a čas dodania je 1 hodín. Dodávateľ 4 bude dodávať 410 jednotiek tovaru zákazníkovi 5. Náklady na jednotku sú 3 € a čas dodania je 1 hodín. Celkové náklady na prepravu dosahujú sumu 5760 €.

Obr. 4.1.6 Tabuľka a interpretácia riešenia dopravnej úlohy
Zdroj: vlastné spracovanie



Obr. 4.1.5 Vizualizácia riešenia dopravnej úlohy - tepelná mapa
Zdroj: vlastné spracovanie



Obr. 4.1.4 Vizualizácia riešenia dopravnej úlohy
Zdroj: vlastné spracovanie

```
library(lpSolve)
n=5
m=4
C <- matrix(c(7,4,2,8,10,
              5,3,9,9,4,
              12,5,7,4,2,
              8,8,6,11,3),nrow=m,byrow=TRUE)
dod <- rep("<=",m)
kapacita <- c(620,400,350,545)
odb <- rep("=",n)
dopyt <- c(235,300,180,420,410)
lp.transport(C,"min",dod,kapacita,odb,dopyt)
lp.transport(C,"min",dod,kapacita,odb,dopyt)$solution
```

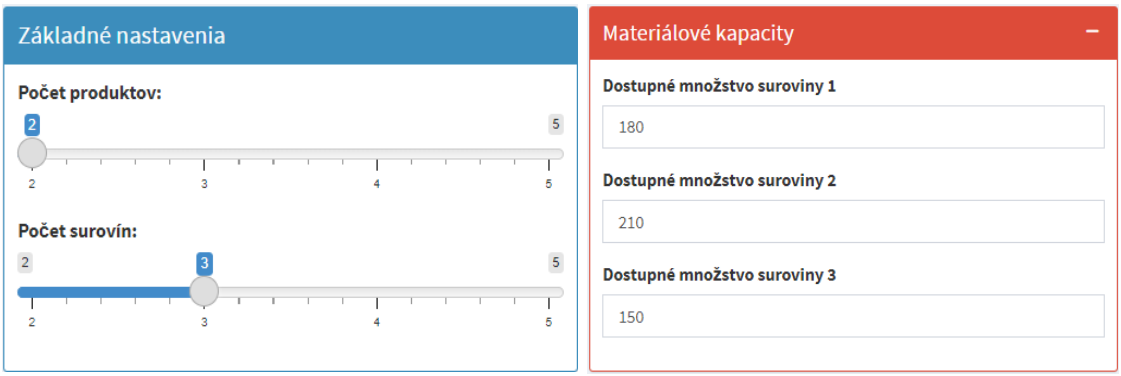
Obr. 4.1.7 Kontrola riešenia dopravnej úlohy pomocou kódu v R
Zdroj: vlastné spracovanie

```
> lp.transport(C,"min",dod,kapacita,odb,dopyt)
Success: the objective function is 5760
> lp.transport(C,"min",dod,kapacita,odb,dopyt)$solution
     [,1] [,2] [,3] [,4] [,5]
[1,]    0  135  180   70    0
[2,]  235  165    0    0    0
[3,]    0    0    0  350    0
[4,]    0    0    0    0  410
```

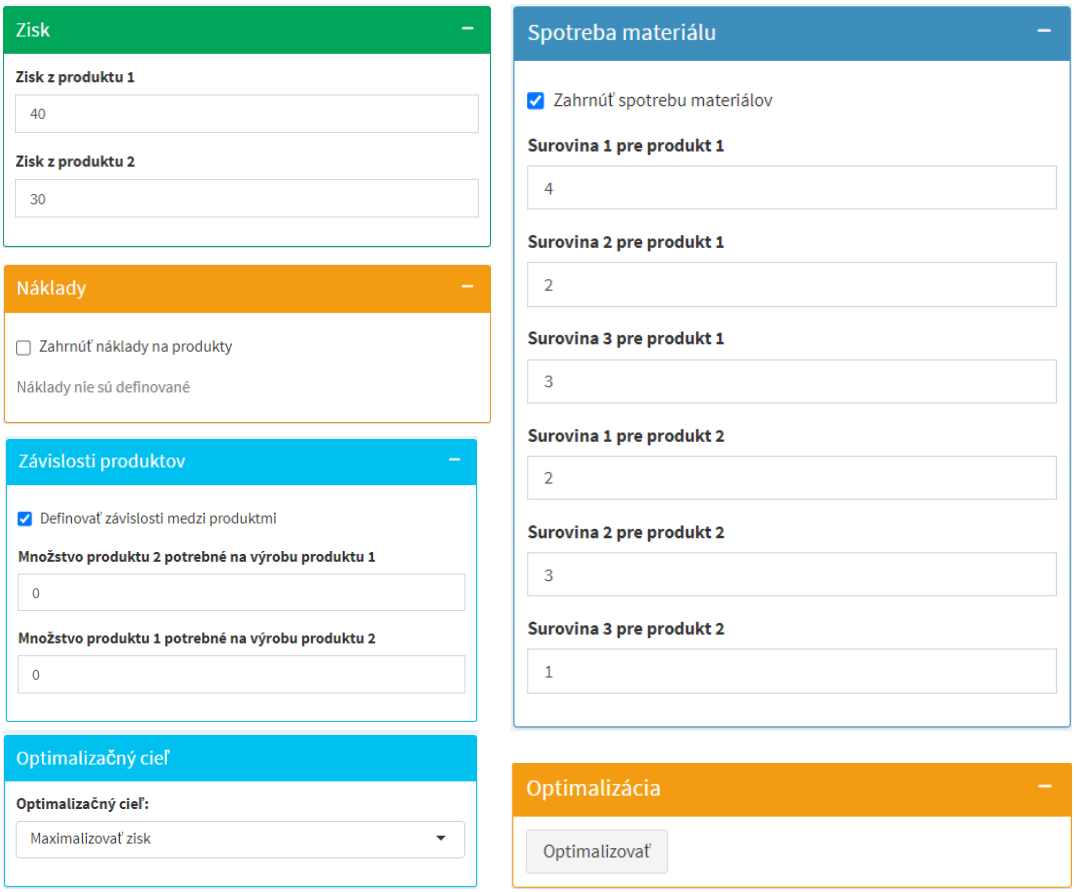
Obr. 4.1.8 Výsledok riešenia dopravnej úlohy v R konzole
Zdroj: vlastné spracovanie

4.1.2 Aplikácia: Výrobný problém

Na overenie správnosti výpočtu aplikácie pre výrobný problém využijeme formuláciu príkladu z časti 3.2.1, kde firma vyrábala 2 druhy produktov (sviečky v tvare srdca a kruhu) s danými obmedzeniami a ziskami. Tieto vstupné údaje vložíme do aplikácie.

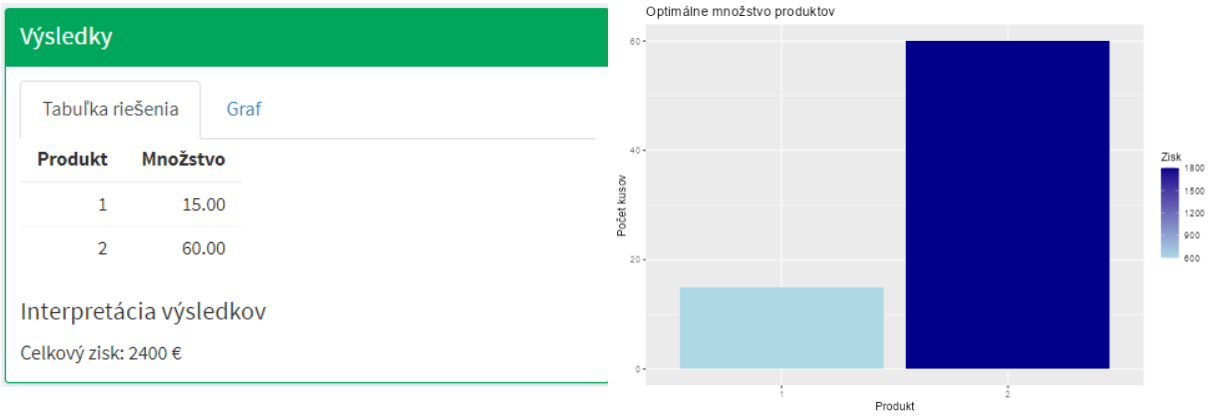


Obr. 4.1.9 Zadávanie vstupných údajov pre výrobný problém
Zdroj: vlastné spracovanie



Obr. 4.1.11 Zadávanie spotreby a optimalizačného cieľa výrobný problém
Zdroj: vlastné spracovanie

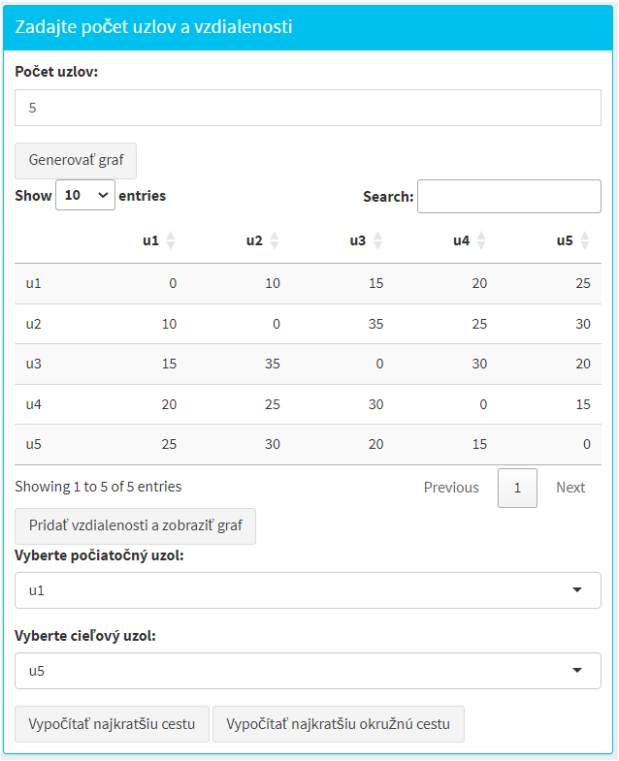
Aplikácia po zadaní príslušných parametrov vrátila optimálny výrobný plán aj celkový zisk, ktoré sa presne zhodujú s manuálnym výpočtom uvedeným v príklade 3.2.1, čo potvrdzuje správnosť implementácie optimalizačného algoritmu.



Obr. 4.1.10 Tabuľka, interpretácia, a graf riešenia výrobného problému
Zdroj: vlastné spracovanie

4.1.3 Aplikácia: Hľadanie najkratšej a najkratšej okružnej cesty

Najskôr si vygenerujeme tabuľku pre 5 uzlov, do ktorej vložíme príslušné vzdialenosti podľa tab. 3.2.4. Následne vytvoríme orientovaný graf s týmito hodnotami a po zadaní počiatočného a cieľového uzla zistíme najkratšiu cestu. Okrem toho vypočítame aj najkratšiu okružnú cestu.



Obr. 4.1.12 Vstupná tabuľka vzdialeností uzlov
Zdroj: vlastné spracovanie

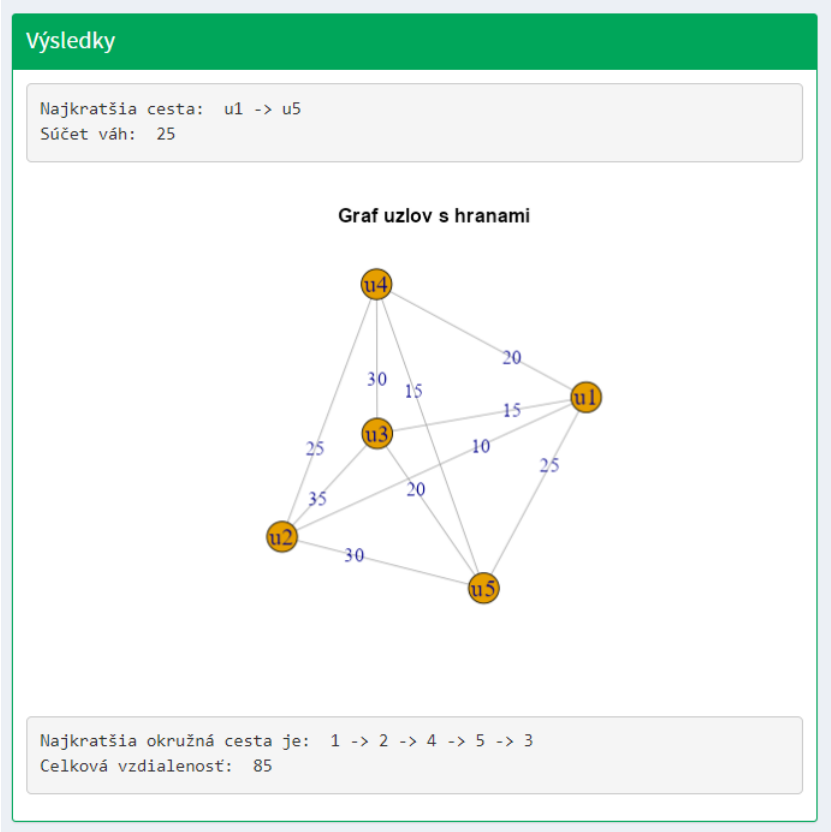
```
vzdialenosti <- matrix(c(
  0, 10, 15, 20, 25,
  10, 0, 35, 25, 30,
  15, 35, 0, 30, 20,
  20, 25, 30, 0, 15,
  25, 30, 20, 15, 0
), nrow = 5, byrow = TRUE)

cat("Najkratšia vzdialenosť z u1 do u5 je:", vzdialenosti[1,5], "\n")
```

Obr. 4.1.13 Kód vstupných údajov pre kontrolu v R
Zdroj: vlastné spracovanie

Keďže v časti 3.2.4 sme pre určené vzdialenosti počítali iba okružnú cestu, dodatočne overíme aj správnosť výpočtu najkratšej cesty z uzla *u1* do *u5* pomocou jazyka R. Porovnaním

výsledkov zisťujeme, že sa opäť zhodujú, čo potvrdzuje správnosť a úspešnú implementáciu danej funkcionality.



Obr. 4.1.14 Vygenerovaný graf a riešenie úloh najkratšej cesty a okružnej cesty
Zdroj: vlastné spracovanie

```
> cat("Najkratšia vzdialenosť z u1 do u5 je:", vzdialenosti[1,5], "\n")
Najkratšia vzdialenosť z u1 do u5 je: 25
```

Obr. 4.1.15 Konzolový výstup výpočtu najkratšej cesty v R
Zdroj: vlastné spracovanie

4.2 Výhody a nevýhody vytvorenej aplikácie

Pri tvorbe aplikácie bolo dôležité zhodnotiť jej praktickú využiteľnosť a identifikovať silné aj slabé stránky. V tejto časti sa zameriame na analýzu prínosov a potencionálnych nedostatkov vytvorenej aplikácie.

4.2.1 Výhody

- **Učebná pomôcka** – aplikácia môže slúžiť ako učebná pomôcka pre predmet Management Science, pretože umožňuje študentom experimentovať s rôznymi scenármi optimalizácie a vidieť výsledky v reálnom čase.
- **Možnosť overiť správnosť výpočtov** – študenti môžu použiť aplikáciu na overenie manuálnych výpočtov pri štúdiu.
- **Vizualizácia výsledkov** – rôzne vizualizácie uľahčujú pochopenie optimálneho riešenia.
- **Flexibilita a rozšíriteľnosť** – do aplikácie je možné jednoducho pridávať nové úlohy a rozširovať jej funkcionalitu. V budúcnosti môže byť doplnená o ďalšie optimalizačné problémy alebo metódy riešenia.
- **Riešenie viacerých úloh súčasne** – v časti teórie grafov aplikácia umožňuje riešiť hľadanie najkratšej cesty a okružnej trasy bez potreby opätovného zadávania vstupov.
- **Jednoduchosť** – používateľ nemusí poznať jazyk R, stačí mu iba vyplniť vstupy v aplikácii a spustiť výpočet.

4.2.2 Nevýhody

- **Nutnosť správneho zadania vstupov** – používateľ musí rozumieť problematike, aby správne zadal všetky vstupné údaje. Nesprávne hodnoty môžu viesť k chybným alebo nerelevantným výsledkom.
- **Neviditeľný priebeh výpočtu** – aplikácia vráti len výsledok, ale neukazuje ako prebieha samotný výpočet.
- **Obmedzená veľkosť vstupných údajov** – pre veľké dáta môže byť aplikácia pomalá, pretože nie je optimalizovaná na veľké matice.
- **Závislosť na R Shiny** – pre spustenie aplikácie je potrebné mať nainštalovaný jazyk R a potrebné balíčky alebo ju hostovať na Shiny Serveri.

ZÁVER

Cieľom tejto práce bolo vytvoriť interaktívnu aplikáciu v prostredí Shiny, ktorá umožní riešiť a vizualizovať úlohy z vybranej oblasti. V priebehu práce sa potvrdilo, že Shiny aplikácie sú intuitívne na tvorbu a majú logickú štruktúru založenú na oddelení používateľského rozhrania a serverovej logiky.

Porovnanie s dostupnými alternatívami, ako sú Dash v jazyku Python alebo nástroje Business Intelligence, ukázalo, že Shiny vyniká najmä vďaka širokým možnostiam vizualizácie a jednoduchej integrácie s analytickými metódami. Aplikácia sa zameriava na riešenie typických úloh z oblasti Management Science, ako je výrobný problém, dopravná úloha, hľadanie najkratšej a najkratšej okružnej cesty. Pri ich implementácii boli využité knižnice `lpSolve` a `TSP`, ktoré umožnili jednoduché modelovanie a výpočty riešení.

Overenie funkčnosti aplikácie prostredníctvom praktických príkladov uvedených v záverečnej kapitole, ako aj jej porovnanie s výsledkami dostupnými v odbornej literatúre, preukázalo jej správne fungovanie a konzistentnosť výsledkov. Napriek tomu má aplikácia niekoľko nedostatkov, ako napríklad, že neumožňuje spracovanie veľkého množstva dát a vyžaduje aspoň základné porozumenie problematiky zo strany používateľa. Navyše, aplikácia neposkytuje podrobný postup výpočtu, čo však bolo v súlade s jej konceptom, keďže jej cieľom bolo poskytnúť jednoduchý a prehľadný nástroj na riešenie základných optimalizačných problémov.

Najväčším prínosom aplikácie je jej potenciál ako učebnej pomôcky, keďže vďaka intuitívnemu ovládaniu umožňuje používateľovi skúmať rôzne scenáre s odlišnými obmedzeniami, vizualizovať riešenia a tým zlepšiť pochopenie optimalizačných metód. Aplikácia zároveň môže slúžiť aj na overenie správnosti manuálne vypočítaných príkladov. Zdrojový kód aplikácie je priložený k práci, čím je zabezpečená možnosť ďalšieho štúdia a rozvoja aplikácie. Vzhľadom na svoju modularitu môže byť aplikácia v budúcnosti rozšírená o ďalšie optimalizačné metódy a úlohy, čím by sa jej využiteľnosť ešte zvýšila.

ZOZNAM POUŽITEJ LITERATÚRY

ATTALI, Dean. *shinyjs: Easily Improve the User Experience of Your 'Shiny' Apps* [online]. The Comprehensive R Archive Network (CRAN) [cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/shinyjs/index.html

BREZINA, Ivan a PEKÁR, Juraj. *Management science: riešenie optimalizačných úloh prostredníctvom MS EXCEL a jazyka R*. I. Vydanie prvé. Bratislava: Ekonóm, 2024. ISBN 978-80-225-5140-3.

CASTILLO, Dylan. *Develop Data Visualization Interfaces in Python With Dash* [online]. Real Python, 2025 [cit. 18. 04. 2025]. Dostupné z: https://realpython.com/python-dash/

CHENG, Joe. *Running Shiny without a server* [online]. GitHub, 2023 [cit. 24. 01. 2025]. Dostupné z: https://jcheng5.github.io/posit-conf-2023-shinylive/?print-pdf=#/title-slide

CHENG, Joe. *The Past and Future of Shiny* [video]. YouTube, 2022, min. 49.05 – 49.50 [cit. 19. 01. 2025]. Dostupné z: https://www.youtube.com/watch?v=HpqLXB_TnpI

CHENG, Joe; XIE, Yihui; WICKHAM, Hadley. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library* [online]. The Comprehensive R Archive Network (CRAN) [cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/leaflet/index.html

DANTZIG, George Bernard. *Lineárne programovanie a jeho rozvoj*. Vydanie prvé. Bratislava: Slovenské vydavateľstvo technickej literatúry, 1966.

DREAMRS. *shinyWidgets: Custom Inputs Widgets for Shiny Applications* [online]. GitHub [cit. 25. 01. 2025]. Dostupné z: https://github.com/dreamRs/shinyWidgets

JIA, Lihua et al. *Development of interactive biological web applications with R/Shiny*. Briefings in Bioinformatics. 2021, 23(1) [cit. 06. 02. 2025]. Dostupné z: https://doi.org/10.1093/bib/bbab415

KASPRZAK, Peter et al. *Six Years of Shiny in Research - Collaborative Development of Web Tools in R. The R Journal*. 2020, 12(2), s. 25. ISSN 2073-4859.

KSP. Dijkstrov algoritmus [online]. 2020 [cit. 29. 03. 2025]. Dostupné z: https://www.ksp.sk/kucharka/dijkstra/

POSIT. *Arrange Elements – Shiny* [online]. [cit. 25. 03. 2025]. Dostupné z: https://shiny.posit.co/r/layouts/arrange/

POSIT. *Plotly Output for Shiny for Python* [online]. [cit. 25. 01. 2025]. Dostupné z: https://shiny.posit.co/py/components/outputs/plot-plotly/

POSIT, PBC. *Shiny Gallery* [online]. [cit. 06. 02. 2025]. Dostupné z: https://shiny.posit.co/r/gallery/

RADEČIĆ, Dario. *Python Dash vs. R Shiny – Which to Choose in 2021 and Beyond* [online]. Appsilon, 2020 [cit. 26. 01. 2025]. Dostupné z: https://www.appsilon.com/post/dash-vs-shiny

RADEČIĆ, Dario. *R Shiny vs Shiny for Python: What are the Key Differences* [online]. Appsilon, 2022 [cit. 25. 01. 2025]. Dostupné z: https://www.appsilon.com/post/r-shiny-vs-shiny-for-python

RSTUDIO. *DT: A Wrapper of the JavaScript Library 'DataTables'* [online]. [cit. 25. 01. 2025]. Dostupné z: https://rstudio.github.io/DT/

RSTUDIO. *Shiny dashboard: Structure* [online]. GitHub, 2021 [cit. 25. 03. 2025]. Dostupné z: https://rstudio.github.io/shinydashboard/structure.html

RSTUDIO. *Shinydashboard: Create Dashboards with 'Shiny'* [online]. The Comprehensive R Archive Network (CRAN) [cit. 25. 01. 2025]. Dostupné z: https://cran.r-project.org/web/packages/shinydashboard/

WICKHAM, Hadley. *Mastering Shiny.* Reactive building blocks [online]. 2021 [cit. 25. 03. 2025].

WICKHAM, Hadley. *Mastering Shiny.* Why reactivity? [online]. O'Reilly Media 2021 [cit. 24. 01. 2025]. Dostupné z: https://mastering-shiny.org/reactive-motivation.html

PRÍLOHY

Príloha: Kód vytvorenej aplikácie

```
library(shiny)
library(shinydashboard)
library(lpSolve)
library(ggplot2)
library(reshape2)
library(leaflet)
library(dplyr)
library(igraph)
library(DT)
library(TSP)

ui <- dashboardPage(
  skin = "purple",
  dashboardHeader(title = "MSoptimize"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Úvod", tabName = "intro", icon = icon("home")),
      menuItem("Dopravná úloha", tabName = "transport", icon =
icon("truck")),
      menuItem("Výrobný problém", tabName = "vyroba", icon =
icon("industry")),
      menuItem(text = span("Hľadanie ciest", title = "Hľadanie
najkratšej a najkratšej okružnej cesty"), tabName = "cesty", icon =
icon("project-diagram"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "intro",
              h2("Vitajte v aplikácii MSoptimize"),
              fluidRow(
                box(width = 8, background = "purple",
                    p("Túto aplikáciu môžete použiť na riešenie a
vizualizáciu výsledkov troch optimalizačných problémov:"),
                    tags$ul(
                      tags$li(tags$b("Dopravná úloha:"), "
Optimalizácia distribúcie tovaru medzi dodávateľmi a zákazníkmi."),
                      tags$li(tags$b("Výrobný problém:"), "
Optimalizácia výrobných procesov a alokácie zdrojov."),
                      tags$li(tags$b("Hľadanie najkratšej a
najkratšej okružnej cesty."))
                    ),
                    br(),
                    p("Aplikácia poskytuje:"),
                    tags$ul(
                      tags$li("Interaktívne vizualizácie"),
                      tags$li("Možnosť zmeny parametrov v reálnom
čase"),
                      tags$li("Detailné výsledky optimalizácie")
                    ),
                    br(),
```

```
                    p(tags$i("Na spustenie vyberte požadovanú úlohu v
ľavom menu."))
                )
              )
      ),
      tabItem(tabName = "transport",
              fluidRow(
                box(title = "Základné nastavenia", status = "primary",
solidHeader = TRUE, width = 12,
                    sliderInput("n_suppliers", "Počet dodávateľov:",
min = 2, max = 5, value = 3),
                    sliderInput("n_customers", "Počet zákazníkov:",
min = 2, max = 5, value = 3)
                ),
              ),
              fluidRow(
                box(title = "Kapacity dodávateľov", status = "info",
solidHeader = TRUE, width = 6,
                    uiOutput("supplier_capacity")
                ),
                box(title = "Dopyty zákazníkov", status = "danger",
solidHeader = TRUE, width = 6,
                    uiOutput("customer_demand"))
              ),
              fluidRow(
                box(title = "Náklady", status = "success", solidHeader
= TRUE, width = 6,
                    uiOutput("cost_matrix")
                ),
                box(title = "Časy", status = "success", solidHeader =
TRUE, width = 6,
                    uiOutput("time_matrix"),
                ),
              ),
              fluidRow(
                box(title = "Dodatočné obmedzenia", status =
"warning", solidHeader = TRUE, width = 6,
                    checkboxInput("limit_time", "Obmedziť maximálnu
dobu doručenia?", value = FALSE),
                    conditionalPanel(condition = "input.limit_time ==
true",
                                     numericInput("max_time",
"Maximálna doba doručenia (v hodinách):", value = 1)
                    ),
                    checkboxInput("limit_capacity", "Obmedziť
kapacitu vozidiel?", value = FALSE),
                    conditionalPanel(condition =
"input.limit_capacity == true",
                                     numericInput("vehicle_capacity",
"Kapacita vozidla (v tonách):", value = 1)
                    ),
                ),
```

```
                box(title = "Optimalizačný scenár", status = "info",
solidHeader = TRUE, width = 6,
                    radioButtons("scenario", "Optimalizačný scenár:",
                                 choices = c("Minimálne náklady",
"Minimálny čas", "Kombinácia nákladov a času"),
                                 selected = "Minimálne náklady"),
                    actionButton("optimize_t", "Optimalizovať"),
                )
              ),
              fluidRow(
                box(title = "Výsledky", status = "success",
solidHeader = TRUE, width = 12,
                    tabsetPanel(
                      tabPanel("Tabuľka riešenia",
tableOutput("solution_t")),
                      tabPanel("Graf", plotOutput("transport_plot")),
                      tabPanel("Približné znázornenie prepravných
trás", leafletOutput("transport_map"))
                    ),
                    h4("Interpretácia výsledkov"),
                    textOutput("interpretation")
                )
              )
      ),
      tabItem(
        tabName = "vyroba",
        fluidRow(
          box(title = "Základné nastavenia", width = 12, solidHeader
= TRUE, status = "primary",
              sliderInput("num_products", "Počet produktov:", min = 2,
max = 5, value = 3),
              sliderInput("num_materials", "Počet surovín:", min = 2,
max = 5, value = 3)
          )
        ),

        fluidRow(
          box(title = "Zisk", width = 6, collapsible = TRUE,
solidHeader = TRUE, status = "success",
              uiOutput("product_profit")
          ),

          box(title = "Náklady", width = 6, collapsible = TRUE,
solidHeader = TRUE, status = "warning",
              checkboxInput("include_costs", "Zahrnúť náklady na
produkty", value = FALSE),
              uiOutput("production_cost"),
              conditionalPanel(
                condition = "!input.include_costs",
                helpText("Náklady nie sú definované")
              )
          )
```

```
        ),
        fluidRow(
          box(title = "Materiálové kapacity", width = 12, collapsible
= TRUE, solidHeader = TRUE, status = "danger",
              uiOutput("material_availability"),
          )
        ),

        fluidRow(
          box(title = "Spotreba materiálu", width = 6, collapsible =
TRUE, solidHeader = TRUE, status = "primary",
              checkboxInput("include_materials",   "Zahrnúť   spotrebu
materiálov", value = TRUE),
              uiOutput("material_usage")
          ),

          box(title = "Závislosti produktov", width = 6, collapsible
= TRUE, solidHeader = TRUE, status = "info",
              checkboxInput("dependency_checkbox",          "Definovať
závislosti medzi produktmi", value = TRUE),
              uiOutput("product_dependencies"),
              conditionalPanel(
                condition = "!input.dependency_checkbox",
                helpText("Závislosti produktov nie sú definované")
              )
          )
        ),

        fluidRow(
          box(title = "Optimalizačný cieľ", width = 6, solidHeader =
TRUE, status = "info",
              selectInput("objective", "Optimalizačný cieľ:",
                          choices    =    c("Maximalizovať    zisk"    =
"max_profit",
                                           "Minimalizovať    náklady"    =
"min_cost"))
          ),
          box(title = "Optimalizácia", width = 6, collapsible = TRUE,
solidHeader = TRUE, status = "warning",
              actionButton("optimize_v", "Optimalizovať")
          )
        ),

        fluidRow(
          box(title = "Výsledky", status = "success", solidHeader =
TRUE, width = 12,
              tabsetPanel(
                tabPanel("Tabuľka                                riešenia",
tableOutput("solution_v")),
                tabPanel("Graf", plotOutput("solution_plot"))
              ),
              h4("Interpretácia výsledkov"),
```

```
            textOutput("total_profit_or_cost")
          )
        )
      ),
      tabItem(
        tabName = "cesty",
        fluidRow(
          box(title = "Zadajte počet uzlov a vzdialenosti", status =
"info", solidHeader = TRUE, width = 5,
            numericInput("num_nodes", "Počet uzlov:", 5, min = 2,
max = 10),
            actionButton("generate_graph", "Generovať graf"),
            DTOutput("distance_table"),
            actionButton("add_distances", "Pridať vzdialenosti a
zobraziť graf"),
            selectInput("dijkstra_from", "Vyberte počiatočný
uzol:", choices = NULL),
            selectInput("dijkstra_to", "Vyberte cieľový uzol:",
choices = NULL),
            actionButton("calculate_dijkstra", "Vypočítať
najkratšiu cestu"),
            actionButton("calculate_tsp", "Vypočítať najkratšiu
okružnú cestu")
          ),
          box(title = "Výsledky", status = "success", solidHeader =
TRUE, width = 7,
            verbatimTextOutput("dijkstra_result"),
            plotOutput("graph_plot"),
            verbatimTextOutput("tsp_result")
          )
        )
      )
    )
  )
)

server <- function(input, output, session) {

  #---------------------Dopravná úloha-------------------------
  output$supplier_capacity <- renderUI({
    lapply(1:input$n_suppliers, function(i) {
      numericInput(paste0("capacity_", i), paste("Kapacita
dodávateľa", i), value = 10)
    })
  })
  output$customer_demand <- renderUI({
    lapply(1:input$n_customers, function(i) {
      numericInput(paste0("demand_", i), paste("Dopyt zákazníka",
i),value = 10)
    })
  })
  output$cost_matrix <- renderUI({
```

```
    lapply(1:input$n_suppliers, function(i) {
      lapply(1:input$n_customers, function(j) {
        numericInput(
          inputId = paste0("cost_", i, "_", j),
          label = HTML(paste0(
            "Cena za jednotku (Dodávateľ ", i,
            " <i class='fas fa-arrow-right' style='margin: 0 5px;
color: #555;'></i> ",
            "Zákazník ", j, "):"
          )),
          value = 10,
          min = 0
        )
      })
    })
  })
  output$time_matrix <- renderUI({
    lapply(1:input$n_suppliers, function(i) {
      lapply(1:input$n_customers, function(j) {
        numericInput(
          inputId = paste0("time_", i, "_", j),
          label = HTML(paste0(
            "Čas prepravy (Dodávateľ ", i,
            " <i class='fas fa-arrow-right' style='margin: 0 5px;
color: #555;'></i> ",
            "Zákazník ", j, "):"
          )),
          value = 1,
          min = 0
        )
      })
    })
  })

  observeEvent(input$optimize_t, {
    num_suppliers <- input$n_suppliers
    num_customers <- input$n_customers
    supplier_capacity <- sapply(1:num_suppliers, function(i) {
      input[[paste0("capacity_", i)]] %||% 0
    })
    customer_demand <- sapply(1:num_customers, function(i) {
      input[[paste0("demand_", i)]] %||% 0
    })
    cost_matrix <- matrix(0, nrow = num_suppliers, ncol =
num_customers)
    time_matrix <- matrix(0, nrow = num_suppliers, ncol =
num_customers)

    for (i in 1:num_suppliers) {
      for (j in 1:num_customers) {
        cost_matrix[i, j] <- input[[paste0("cost_", i, "_", j)]] %||%
0
```

```
        time_matrix[i, j] <- input[[paste0("time_", i, "_", j)]] %||%
0
      }
    }
    if (input$limit_time) {
      max_time <- input$max_time
      time_matrix[time_matrix > max_time] <- Inf
      cost_matrix[time_matrix > max_time] <- Inf
    }
    optimization_matrix <- switch(input$scenario,
                                  "Minimálne náklady" = cost_matrix,
                                  "Minimálny čas" = time_matrix,
                                  "Kombinácia nákladov a času" =
0.5*cost_matrix + 0.5*time_matrix)
    total_supply <- sum(supplier_capacity)
    total_demand <- sum(customer_demand)

    if (total_supply < total_demand) {
      showNotification("Celková kapacita dodávateľov je menšia ako
celkový dopyt!", type = "error")
      return()
    }

    result <- lp.transport(
      cost.mat = optimization_matrix,
      direction = "min",
      row.signs = rep("<=", num_suppliers),
      row.rhs = supplier_capacity,
      col.signs = rep("=", num_customers),
      col.rhs = customer_demand,
      integers = NULL
    )

    if (result$status != 0) {
      showNotification("Optimalizácia zlyhala - problém nemá
riešenie", type = "error")
      return()
    }

    solution_matrix <- matrix(result$solution, nrow = num_suppliers,
ncol = num_customers)
    output$solution_t <- renderTable({
      df <- as.data.frame(solution_matrix)
      names(df) <- paste("Zákazník", 1:num_customers)
      rownames(df) <- paste("Dodávateľ", 1:num_suppliers)
      df
    }, rownames = TRUE)
    total_cost <- sum(solution_matrix * cost_matrix)
    output$interpretation <- renderText({
      paste("Celkové optimálne náklady na prepravu sú",
round(total_cost, 2), "eur.")
    })
```

```
    output$transport_plot <- renderPlot({
      transport_df <- melt(solution_matrix)
      colnames(transport_df)     <-     c("Supplier",     "Customer",
"Transported")
      ggplot(transport_df,    aes(x    =    factor(Supplier),    y    =
factor(Customer), fill = Transported)) +
        geom_tile() +
        scale_fill_gradient(low = "white", high = "blue") +
        labs(title = "Množstvo prepraveného tovaru", x = "Dodávateľ",
y = "Zákazník") +
        theme_minimal()
    })
    output$transport_map <- renderLeaflet({

      supplier_coords  <-  data.frame(lat  =  runif(num_suppliers,  48,
50),
                                      lng  =  runif(num_suppliers,  16,
19))
      customer_coords  <-  data.frame(lat  =  runif(num_customers,  48,
50),
                                      lng  =  runif(num_customers,  16,
19))
      colors    <-    colorFactor(palette    =    "Set1",    domain    =
1:num_suppliers)

      map <- leaflet() %>% addTiles()

      for (i in 1:num_suppliers) {
        map <- map %>%
          addMarkers(lng = supplier_coords$lng[i],
                     lat = supplier_coords$lat[i],
                     popup = paste("Dodávateľ", i))
      }

      for (j in 1:num_customers) {
        map <- map %>%
          addMarkers(lng = customer_coords$lng[j],
                     lat = customer_coords$lat[j],
                     popup = paste("Zákazník", j))
      }

      for (i in 1:num_suppliers) {
        for (j in 1:num_customers) {
          if (solution_matrix[i, j] > 0) {
            map <- map %>%
              addPolylines(
                lng            =            c(supplier_coords$lng[i],
customer_coords$lng[j]),
                lat            =            c(supplier_coords$lat[i],
customer_coords$lat[j]),
                color = colors(i),
```

```
                weight = solution_matrix[i, j] / max(solution_matrix)
* 5,
                popup = paste("Množstvo:", solution_matrix[i, j],
                              "<br>Dodávateľ:", i,
                              "<br>Zákazník:", j),
                label = paste("Množstvo:", solution_matrix[i, j]),
                layerId = paste0("link_", i, "_", j)
              )
          }
        }
      }

      map
    })

    output$interpretation <- renderText({

      total_cost <- sum(solution_matrix * cost_matrix)
      total_time <- sum(solution_matrix * time_matrix)

      transport_details <- ""
      for(i in 1:num_suppliers) {
        for(j in 1:num_customers) {
          if(solution_matrix[i,j] > 0) {
            transport_details <- paste0(transport_details,
                                        "Dodávateľ ", i, " bude
dodávať ", round(solution_matrix[i,j], 2),
                                        "      jednotiek      tovaru
zákazníkovi ", j, ". ",
                                        "Náklady na jednotku sú ",
round(cost_matrix[i,j], 2), " € ",
                                        "a   čas   dodania   je   ",
round(time_matrix[i,j], 2), " hodín.\n\n")
          }
        }
      }
       if(input$scenario == "Minimálne náklady") {
        scenario_text <- paste("Celkové náklady na prepravu dosahujú
sumu ", round(total_cost, 2), " €.")
      } else if(input$scenario == "Minimálny čas") {
        scenario_text <- paste("Celkový čas potrebný na prepravu je ",
round(total_time, 2), " hodín.")
      } else {
        scenario_text <- paste("Pri kombinovanom hodnotení sú celkové
náklady ", round(total_cost, 2),
                               "  €  a  celkový  čas  prepravy  ",
round(total_time, 2), " hodín.")
      }

      paste("Optimálny prepravný plán:\n\n",
            transport_details, "\n",
            scenario_text)
```

```
    })
  })
  #--------------------Výrobný problém-----------------------------
  output$product_profit <- renderUI({
    lapply(1:input$num_products, function(i) {
      numericInput(paste0("profit_", i), paste("Zisk z produktu", i),
value = 10, min = 0)
    }) %>% tagList()
  })
  output$product_cost <- renderUI({
    lapply(1:input$num_products, function(i) {
      numericInput(paste0("production_cost_", i), paste("Náklady na
produkt", i), value = 5, min = 0)
    }) %>% tagList()
  })
  output$material_usage <- renderUI({
    lapply(1:input$num_products, function(i) {
      lapply(1:input$num_materials, function(j) {
        numericInput(paste0("material_", i, "_", j),
                     paste("Surovina", j, "pre produkt", i),
                     value = 5, min = 0)
      })
    }) %>% tagList()
  })
  output$material_availability <- renderUI({
    lapply(1:input$num_materials, function(j) {
      numericInput(paste0("availability_", j),
                   paste("Dostupné množstvo suroviny", j),
                   value = 100, min = 0)
    }) %>% tagList()
  })
  output$product_dependencies <- renderUI({
    if (input$dependency_checkbox) {
      lapply(1:input$num_products, function(i) {
        lapply(1:input$num_products, function(j) {
          if (i != j) {
            numericInput(paste0("dependency_", i, "_", j),
                         paste("Množstvo produktu", j, "potrebné na
výrobu produktu", i),
                         value = 0, min = 0)
          }
        })
      }) %>% tagList()
    }
  })
  observeEvent(input$optimize_v, {
    num_products <- input$num_products
    num_materials <- input$num_materials
    profit        <-        sapply(1:num_products,        function(i)
input[[paste0("profit_", i)]] %||% 0)
    cost        <-        sapply(1:num_products,        function(i)
input[[paste0("production_cost_", i)]] %||% 0)
```

```
    material_constraints <- if (num_materials > 0) {
      mat <- matrix(0, nrow = num_materials, ncol = num_products)
      for (j in 1:num_materials) {
        for (i in 1:num_products) {
          mat[j, i] <- input[[paste0("material_", i, "_", j)]] %||% 0
        }
      }
      mat
    } else {
      matrix(0, nrow = 0, ncol = num_products)
    }
    material_availability <- if (num_materials > 0) {
      sapply(1:num_materials,                                  function(j)
input[[paste0("availability_", j)]] %||% Inf)
    } else numeric(0)
    dependency_constraints   <-   if   (input$dependency_checkbox   &&
num_products > 1) {
      mat <- matrix(0, nrow = num_products, ncol = num_products)
      for (i in 1:num_products) {
        for (j in 1:num_products) {
          if (i != j) mat[i, j] <- input[[paste0("dependency_", i,
"_", j)]] %||% 0
        }
      }
      mat
    } else {
      matrix(0, nrow = 0, ncol = num_products)
    }
    all_constraints <- matrix(numeric(0), ncol = num_products)
    all_directions <- character(0)
    all_rhs <- numeric(0)
    if (nrow(material_constraints) > 0) {
      all_constraints <- rbind(all_constraints, material_constraints)
      all_directions       <-       c(all_directions,       rep("<=",
nrow(material_constraints)))
      all_rhs <- c(all_rhs, material_availability)
    }
    if (nrow(dependency_constraints) > 0) {

      dep_matrix <- diag(num_products) - dependency_constraints
      all_constraints <- rbind(all_constraints, dep_matrix)
      all_directions <- c(all_directions, rep(">=", num_products))
      all_rhs <- c(all_rhs, rep(0, num_products))
    }
    objective_function <- if (input$objective == "max_profit") profit
else -cost

    result <- tryCatch({
      if (nrow(all_constraints) == 0) {

        list(status  =  0,  solution  =  rep(ifelse(input$objective  ==
"max_profit", Inf, 0), num_products))
```

```
      } else {
        lp_result <- lp(
          direction = "max",
          objective.in = objective_function,
          const.mat = all_constraints,
          const.dir = all_directions,
          const.rhs = all_rhs,
          all.int = TRUE
        )
        if (lp_result$status == 0) {
          list(status = 0, solution = lp_result$solution)
        } else {
          list(status = lp_result$status, message = "No feasible
solution")
        }
      }
    }, error = function(e) {
      list(status = 2, message = paste("Optimization error:",
e$message))
    })


    if (result$status == 0) {
      output$solution_v <- renderTable({
        data.frame(
          Produkt = 1:num_products,
          Množstvo       =       ifelse(is.infinite(result$solution),
"Unlimited", round(result$solution, 2))
        )
      })

      output$total_profit_or_cost <- renderText({
        if (any(is.infinite(result$solution))) {
          if (input$objective == "max_profit") "Unlimited profit
possible" else "Zero cost possible"
        } else if (input$objective == "max_profit") {
          paste("Celkový zisk:", round(sum(result$solution * profit),
2), "€")
        } else {
          paste("Celkové náklady:", round(sum(result$solution * cost),
2), "€")
        }
      })

      output$solution_plot <- renderPlot({
        if (!any(is.infinite(result$solution))) {
          data <- data.frame(
            Produkt = factor(1:num_products),
            Množstvo = result$solution,
            Zisk = result$solution * profit
          )
          ggplot(data, aes(x = Produkt, y = Množstvo, fill = Zisk)) +
```

```
            geom_col() +
            scale_fill_gradient(low = "lightblue", high = "darkblue")
+
            labs(title = "Optimálne množstvo produktov", y = "Počet
kusov")
        }
      })
    } else {
      output$total_profit_or_cost <- renderText({
        paste("Chyba optimalizácie:", result$message %||% "Neznáma
chyba")
      })
      output$solution_v <- renderTable(NULL)
      output$solution_plot <- renderPlot(NULL)
    }
  })

  #---------------Hľadanie ciest---------------------------
  nodes <- reactiveVal(data.frame(id = character(0)))
  distances <- reactiveVal(matrix(NA, nrow = 0, ncol = 0))

  observeEvent(input$generate_graph, {
    num_nodes <- input$num_nodes
    new_nodes <- data.frame(id = paste0("u", 1:num_nodes))
    nodes(new_nodes)

    dist_matrix <- matrix(NA, nrow = num_nodes, ncol = num_nodes)
    rownames(dist_matrix) <- new_nodes$id
    colnames(dist_matrix) <- new_nodes$id
    diag(dist_matrix) <- 0
    distances(dist_matrix)

    output$distance_table <- renderDT({
      datatable(distances(), editable = TRUE, options =
list(pageLength = 10, scrollX = TRUE))
    })

    updateSelectInput(session, "dijkstra_from", choices =
new_nodes$id)
    updateSelectInput(session, "dijkstra_to", choices = new_nodes$id)
  })

  observeEvent(input$distance_table_cell_edit, {
    info <- input$distance_table_cell_edit
    i <- info$row
    j <- info$col
    new_value <- suppressWarnings(as.numeric(info$value))
    if (!is.na(new_value) && new_value >= 0) {
      dist_matrix <- distances()
      dist_matrix[i, j] <- new_value
      dist_matrix[j, i] <- new_value
      diag(dist_matrix) <- 0
```

```
      distances(dist_matrix)
    }
  })

  observeEvent(input$add_distances, {
    dist_matrix <- distances()

    edge_list <- which(upper.tri(dist_matrix) & !is.na(dist_matrix) &
dist_matrix > 0, arr.ind = TRUE)
    edges <- data.frame(
      from = rownames(dist_matrix)[edge_list[, 1]],
      to = colnames(dist_matrix)[edge_list[, 2]],
      weight = dist_matrix[edge_list]
    )

    net <- graph_from_data_frame(edges, vertices = nodes(), directed
= FALSE)

    output$graph_plot <- renderPlot({
      plot(
        net,
        main = "Graf uzlov s hranami",
        vertex.size = 20,
        vertex.label.cex = 1.5,
        edge.label = round(E(net)$weight, 2),
        edge.label.cex = 1.2
      )
    })
  })
  observeEvent(input$calculate_dijkstra, {
    from <- input$dijkstra_from
    to <- input$dijkstra_to
    dist_matrix <- distances()
    net    <-    graph_from_adjacency_matrix(dist_matrix,    mode    =
"undirected", weighted = TRUE)

    shortest_path_result <- tryCatch({
      shortest_paths(net, from = from, to = to, algorithm = "dijkstra")
    }, error = function(e) NULL)

    output$dijkstra_result <- renderText({
      if (is.null(shortest_path_result)) {
        "Chyba pri výpočte najkratšej cesty. Skontrolujte vstupy."
      } else {
        path_nodes <- shortest_path_result$vpath[[1]]
        if (length(path_nodes) > 0) {
          path_edges <- E(net, path = path_nodes)
          total_weight <- sum(E(net)$weight[path_edges])
          paste("Najkratšia cesta: ", paste(V(net)$name[path_nodes],
collapse = " -> "),
                "\nSúčet váh: ", total_weight)
        } else {
```

```
          paste("Cesta medzi uzlami", from, "a", to, "neexistuje.")
        }
      }
    })
  })

  observeEvent(input$calculate_tsp, {
    dist_matrix <- distances()

    dist_matrix[dist_matrix == 0 & row(dist_matrix) != col(dist_matrix)] <- Inf

    net <- graph_from_adjacency_matrix(dist_matrix, mode = "undirected", weighted = TRUE)
    if (!is_connected(net)) {
      output$tsp_result <- renderText({
        "Graf nie je súvislý. Nie je možné nájsť okružnú cestu."
      })
      return()
    }

    atsp <- ATSP(dist_matrix)
    tour <- solve_TSP(atsp, method = "nn", start = 1)

    output$tsp_result <- renderText({
      tour_nodes <- tour[]
      total_distance <- sum(
        dist_matrix[cbind(tour_nodes, c(tour_nodes[-1], tour_nodes[1]))], na.rm = TRUE
      )
      paste("Najkratšia okružná cesta je: ", paste(tour_nodes, collapse = " -> "),
            "\nCelková vzdialenosť: ", total_distance)
    })
  })
}

shinyApp(ui, server)
```