VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPTIMÁLNÍ ŘÍZENÍ SVĚTELNÉ KŘIŽOVATKY

OPTIMAL CONTROL OF INTERSECTION SIGNALING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Petr Tinka
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jakub Kůdela, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Petr Tinka**
Studijní program:	Aplikovaná informatika a řízeníStudijní obor: bez specializace
Vedoucí práce:	**Ing. Jakub Kůdela, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijníma zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimální řízení světelné křižovatky

1 Stručná charakteristika problematiky úkolu:

Moderní metody pro efektivní řízení městské dopravy využívají různé typy formulací a modelů pro řízení světelných křižovatek. Mezi tyto typy patří například vícekriteriální modely, nebo kooperativní modely pro řízení více křižovatek najednou. Diplomová práce se bude zabývat optimálním řízením světelné křižovatky pro vybranou moderní formulaci. Hlavním cílem práce pak bude implementace některé z aktuálně používaných method.

2 Cíle diplomové práce:

Popsat problém řízení světelné křížovatky.
Provést rešerši aktuálně používaných metod pro tento problém.Pro vybranou metodu vytvořit software implementaci.

3 Seznam doporučené literatury:

LI, X., SUN, J.-Q. Turning-lane and signal optimization at intersections with multiple objectives.Engineering Optimization. 2019, 51 (3), 484-502.

LIST, G. F., CETIN, M. Modeling Traffic Signal Control Using Petri Nets. IEEE Transactions onIntelligent Transportation Systems. 2004, 5 (3), 177-187.

CHEN, L., ENGLUND, C. Cooperative Intersection Management: A Survey. IEEE Transactions onIntelligent Transportation Systems. 2016, 17 (2), 570-586.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.	doc. Ing. Jiří Hlinka, Ph.D.
ředitel ústavu	děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá řízením světelné křižovatky. V první, teoretické části této práce, probíhá popis tohoto problému. Dále je uveden přehled používaných optimalizačních metod pro tento problém. Praktická část se zabývá vymodelováním sítě světelné křižovatky v programu SUMO a na ni implementací vybraných optimalizačních metod. Na závěr je provedeno porovnání výsledků.

ABSTRACT

This master´s thesis is about the control of a traffic light. In the first, theoretical part of this work, a description of this problem introduce. The following is an overview of the optimization methods used for this problem. The practical part deals with the modeling of the network of traffic lights in program SUMO and the implementation of selected optimization methods on this. Finally, a comparison of results is made.

KLÍČOVÁ SLOVA

Optimalizace světelné křižovatky, dopravní síť, SUMO, genetický algoritmus, Powell, Nelder-Mead.

KEYWORDS

Crossroads optimization, transport network, SUMO, genetic algorithm, Powell, Nelder-Mead.

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

TINKA, Petr. *Optimální řízení světelné křižovatky*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022, 65 s. Diplomová práce. Vedoucí práce: Ing. Jakub Kůdela, Ph.D.

PODĚKOVÁNÍ

Velmi bych chtěl poděkovat vedoucímu této práce Ing. Jakubu Kůdelovi Ph.D. za jeho cenné rady připomínky, a hlavně za čas, který mi vždy ochotně poskytl. Speciální díky bych chtěl věnovat své ženě, která mně během magisterského studia neustále podporovala.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Petr Tinka

OBSAH

1 ÚVOD

Dopravní křižovatka je místo, kde se protínají dopravní trasy, které následně vedou do různých směrů. Dělí se do dvou kategorií, a to signalizované a nesignalizované. Nesignalizované křižovatky mají většinou značky pro jízdu nebo zastavení. Hlavní součástí signalizovaných křižovatek (kterými se tato práce převážně zabývá) je semafor. Ten pomohl zlepšit koordinaci dopravních prostředků, bezpečnost a plynulost dopravy.

I když křižovatky zabírají jen relativně malou část silničního systému, náleží jejich řízení k jedněm z nejnáročnějších problémů v dopravní síti [1]. To jak z hlediska plynulosti dopravy, tak i bezpečnosti provozu. Navíc s dramatickým nárustem počtu vozidel se dopravní zácpy stávají čím dál větším problémem, obvzláště ve větších městech. Zejména v dopravních špičkách se tvoří dlouhé fronty vozidel, což vede k prodloužení doby jízdy. Člověka to navíc ovlivňuje nadměrnou únavou, což může vést k duševním onemocněním a dalším zdravotním problémům. V neposlední řadě má ucpaná křižovatka negativní vliv na životní prostředí, jelikož se do ovzduší dostává velké množství škodlivých látek.

K tomu, aby se s výše uvedenými problémy dopravní systém vyrovnal, jsou potřeba efektivní metodická řešení. Pro nesignalizované křižovatky to znamená dodržování přednosti v jízdě. U světelných křižovatek je důležité zejména přesně nastavit hodnotu časování signálu. Neefektivně nastavené semafory mohou být příčinou zmiňovaných dopravních zácp. Pro optimalizaci to tedy znamená vymodelování křižovatky a dopravní situace, na které jsou následně použity různé přístupy optimalizace.

Cílem práce bylo popsat problematiku řízení světelné křižovatky a provést průzkum používaných optimalizačních metod pro tento problém. Nakonec vytvořit softwarovou implementaci pro zvolenou metodu.

Čtenář se nejprve seznámí se systémem řízení světelné křižovatky a jeho dělení. Dojde k popisu vstupních veličin, jejichž nastavení je závislé na celkové efektivitě provozu světelné křižovatky. Tyto vstupní parametry souvisí se seřízením fází dopravního semaforu (pořadí, délka cyklu apod.). Dále budou charakterizovány výstupní veličiny, tedy parametry, které slouží k optimalizaci. Zde patří například dopravní zpoždění, jízdní rychlost atd.

V další části dojde k popisu optimalizace křižovatky, kde bude rozebrána tvorba simulační sítě a používané optimalizační metody. Mezi takové patří genetický algoritmus, Petriho sítě, Q-učení apod. [2]. Dojde také k představením používaných simulačních nástrojů a podrobnější popis programu, který byl využit pro tuto diplomovou práci.

V závěrečné fázi bude představena k vlastní softwarová implementace tří optimalizačních metod (genetický algoritmus, metoda Powella a Nelderova-Meadova) na konkrétní světelné křižovatce. Nejprve se popíše modelování simulačního prostředí a potom použitých metod. Parametr, který byla snaha optimalizovat, bylo celkové zpoždění odjezdu všech vozidel, kdy šlo o jeho minimalizaci. To se dělo úpravou časování jednotlivých fází pro každý semafor v simulačním prostředí.

Na závěr jsou v diplomové práci porovnány výsledky všech tří použitých algoritmů. Došlo ještě ke srovnání s reálnou situací.

2 ŘÍZENÍ SVĚTELNÉ KŘIŽOVATKY

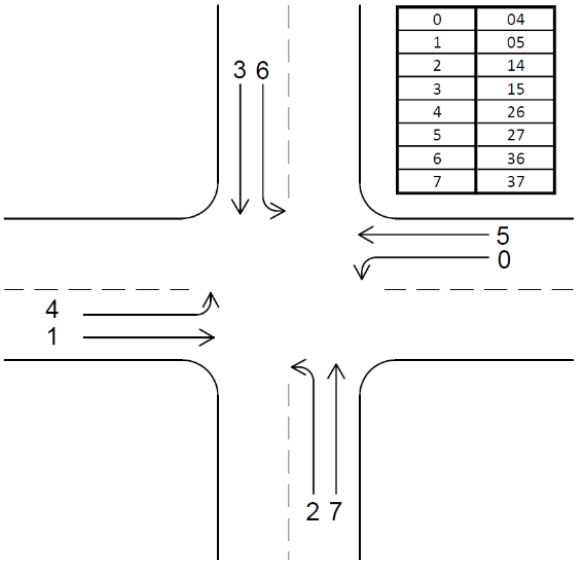
Jak již bylo zmíněno v úvodu, řízení světelné křižovatky je důležité k zajištění bezpečnosti a plynulosti silničního provozu. Špatné nastavení světelného zařízení často vede k dopravním zácpám v horším případě k nehodám. Proto je třeba zajistit co nejlepší možné nastavení parametrů světelné křižovatky.

Všeobecně je známo, že světelná zařízení mají pro vozidla obvykle tří barevnou soustavu, které ještě mohou být doplněny o směrové šipky. Pro chodce mají semafory obyčejně signály dva. Jejich propojení a kombinace na světelné křižovatce potom tvoří fáze, což jsou vlastně časové intervaly, které umožňují bezpečné projetí křižovatkou. Z hlediska řízení je důležité jak načasování jednotlivých fází, jejich délku, posloupnost a podobně. Mezi důležité aspekty, které jsou zohledňovány při nastavování světelného zařízení, patří plynulost dopravy a s tím související doba stání vozidla, bezpečnost provozu, celkový objem provozu jízdní rychlost a jiné.

2.1 Parametry nastavení dopravního signálu

Pro správné plynutí silničního provozu na křižovatkách, je potřeba co nejefektivněji nastavit tyto veličiny světelného zařízení [2]:

- Rozdělení fází,
- pořadí fází,
- fázové přechody,
- mezičas,
- délka cyklu.



Obr.1: Pohyby a fáze jednoduché křižovatky [4]

Fáze se rozdělují tak, aby dopravní pohyby byly co nejvíce bezkolizní anebo podmíněně kolizní, což jsou trasy, pro které platí pravidla přednosti v jízdě podle právního předpisu daného státu [3]. Je zde taky nutné brát v potaz chodce a v případě jeho možného ohrožení by měl být semafor doplněn varovným signálem (obvykle žluté světlo ve tvaru chodce). Z rozdělení následně vyplývá i počet fází. Jejich vyšší počet zvyšuje bezpečnost provozu, avšak prodlužuje zdržení vozidel. Příklad obyčejné křižovatky s minimálním počtem fází je na obrázku 1. Zde první sloupec tabulky udává číslo fáze a ve druhém jsou povolené dopravní pohyby pro každou fázi signálu.

Fázové pořadí bývá ovlivněno různými kritérii. Určité fáze musí probíhat plynule v návaznosti na sebe. Plynulost je zaručena signály „volno" v těchto fázích. Dále musí některé dopravní směry na sebe navazovat, aby nedocházelo k ucpání křižovatky. Pro pořadí je také nutné zohlednit sled doby signálů pro chodce a cyklisty [5].

Koordinovaný čas mezi jednotlivými fázemi neboli mezičas, je interval „žlutého" a „červeného" dopravního signálu, mezi fázemi dopravních signálů. Zajišťuje, aby nedocházelo k protichůdným dopravním pohybům na křižovatce [2]. Správné určení těchto časů je tedy z důvodu bezpečnosti provozu velmi důležité.

Počet stojících vozidel a zpoždění lze snížit zvýšením načasování zelené fáze pro konkrétní pohyb. Ke zvýšení zelené doby jednoho dopravního pohybu však obvykle dochází na úkor zvýšeného zpoždění a počtu stojících vozidel v ostatních dopravních pohybech. Proto je třeba nastavit načasování signálu tak, takový, aby celkový dopravní výkon, např. průměrná doba čekání, byla optimalizována.

2.2 Kategorie řízení

S ohledem na konkrétní křižovatky, jako je jejich stavební stav nebo intenzita dopravy se volí dva základní druhy sytému řízení. A to časově nezávislé nebo časově závislé. V současném stavu zatím platí, že jsou ve světě více použity časově nezávislé systémy řízení [6].

2.2.1 S pevným čase

Stavy dopravy jsou získávány statickou analýzou historických dat intenzity dopravy [6]. Na jejich základě jsou vypočteny výstupní veličiny nutné pro řízení světelné křižovatky. Tato strategie má předem stanovenou délku cyklu. Cílem je minimalizace průměrného zpoždění, maximalizace kapacity apod.

Jsou vhodné především pro dopravní signály, kde je provoz stabilní. Vzhledem k tomu, že zejména městské oblasti jsou mimořádně dynamické, tzn. hrozí časté dopravní kolize, stavební práce a další, které mohou ovlivnit změnu objemu dopravy, jeví se tato strategie v současné době jako nedostatečná.

2.2.2 S dynamickým časem

Cílem této kategorie je optimalizovat dopravu v reálném čase, podle aktuální dopravní situace. Do dopravního uzlu jsou implementovány senzory, které jsou schopné detekovat přítomnost vozidla.

Například tzv. prodlužovací detektor je umístěný cca 30-50 metrů před stop linií a trvale měří rozestupy mezi vozidly. Pokud je tento odstup menší než daná hodnota (obvykle se pohybuje mezi 3-5 s), prodlužuje se délka zelené až do předem daného maxima. Dalším typem detektoru je tzv. výzvový. Ten je umístěn těsně před stop linií nebo v místech, kde se mohou tvořit dopravní zácpy [6]. Pokud je detekována kongesce, do řízení se vkládá nová fáze, která světelnou křižovatku optimalizuje. S vývojem technologií již není sběr dat v reálném čase příliš náročný úkol. Díky využití dostupného výpočetního výkonu je možná okamžitá změna řízení provozu. Nelze však provádět podrobnou analýzu, jako u strategií s pevným časem, protože musí okamžitě regulovat dopravní signály [2].

2.2.3 S adaptivním časem

Velmi se podobá kategorii s dynamickým časem, ale oproti ní využívá předpokládané dopravní podmínky v blízké budoucnosti. Adaptivní strategie bývá implementována pomocí různých predikčních algoritmů [7]. Zmíněné algoritmy však vyžadují vysokou přesnost a dobrou optimalizaci signálního plánu. Proto je vývoj těchto algoritmů obtížnější než u řízení s dynamickým časem. V roce 2017 Lee a kol. [8] se povedlo předpovědět informace včetně odbočování mezi jízdními pruhy, délky front a rychlosti příjezdu. To pomocí rolujícího horizontu, což je metoda snažící se předvídat budoucnost, a poté vypočítání optimalizovaného signálu na základě metody globální optimalizace.

2.3 Parametry pro vyhodnocování

K určení optimální situaci na světelné křižovatce slouží různé veličiny. Volí se takové, které se dají měřit přímo. Podle dopravní situace, která má být optimalizována, se vybírá konkrétní parametr nebo jejich kombinace. Často jsou získávány ze simulace silničního provozu. Vyhodnocují se jak pro všechny účastníky silničního provozu, tak pro jednotlivé dopravní prostředky, či chodce. Mezi parametry vyhodnocování patří:

- průměrná doba čekání vozidel na červený signál,
- propustnost sítě,
- celková doba jízdy,
- spotřeba paliva,
- počet zastavení,
- doba zdržení,
- jízdní rychlost.

Základním parametrem, který se optimalizuje a hledá se jeho minimum, je průměrná doba čekání vozidel na křižovatce na červený signál. Toto měřítko je nejběžnější veličinou pro vyhodnocování dopravní situace [7].

Dalším důležitým aspektem je propustnost sítě. Je definována jako počet vozidel projíždějících sítí. V [9] se Smith aplikoval maximalizaci cestovní kapacity obecné světelné křižovatky, pomocí vhodně zvolené dopravní sítě. Dále se vyhodnocuje celková doba jízdy, což je čas, po který se vozidlo pohybuje v dopravním uzlu. Její minimalizací se docílí plynulejšího provozu dopravních prostředků a úspory času účastníků dopravy [5]. Nebo je snaha o minimalizaci celkového počtu zastávek, průměrného zpoždění nebo celkový počet zastávek.

Jízdní rychlost se získává ke zjištění, zda je možné dosáhnout toho, aby vozidla jela lokálně únosnou rychlostí. Ta je měřena pomocí detektorů umístěných na vhodném místě před světelnou křižovatkou.

V neposlední řadě je v současnosti kladen velký důraz na ochranu životního prostředí. S tím souvisí spotřeba paliva vozidel, počet zastavení a doba zdržení. Ty mají na vliv zvýšení objemu emisí.

3 OPTIMALIZACE SVĚTELNÉ KŘIŽOVATKY

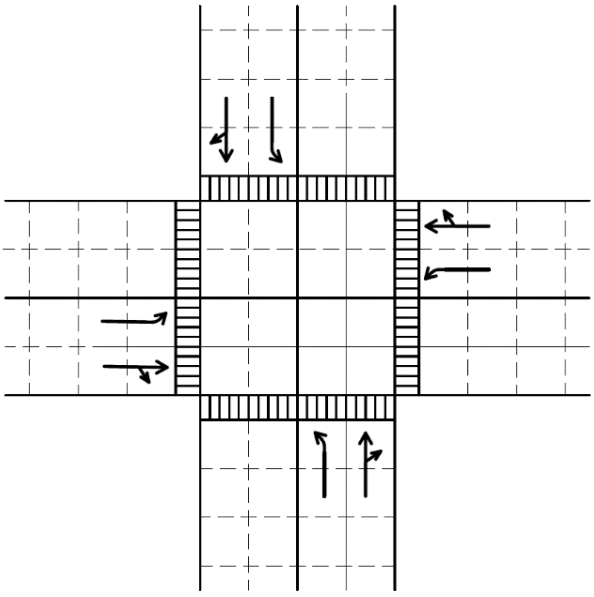
Existuje velké množství přístupů sloužících k optimalizování světelné křižovatky. Jádrem je vytvoření modelu konkrétní křižovatky určené k řešení. Ten je možno vytvořit analyticky, nebo použitím simulačních nástrojů, kterých v dnešní době existuje celá řada. Analytické modely jsou užitečné pro získání náhledu na problém, ale počet interakcí, které je třeba zahrnout do modelu, nebo čas potřebný k řešení, velmi znesnadňují použití těchto analytických modelů. Jejich používání je stále méně časté [2]. Z toho důvodu zde nejsou zahrnuty. Na konkrétní model se poté aplikuje vybraný algoritmus, který minimalizuje, či maximalizuje určený problém (viz výše parametry pro vyhodnocování).

3.1 Modelování křižovatky

Pro optimalizaci světelné křižovatky je základem vymodelování přesné dopravní sítě. Existuje několik přístupů pro její vymodelování.

3.1.1 Diskretizace

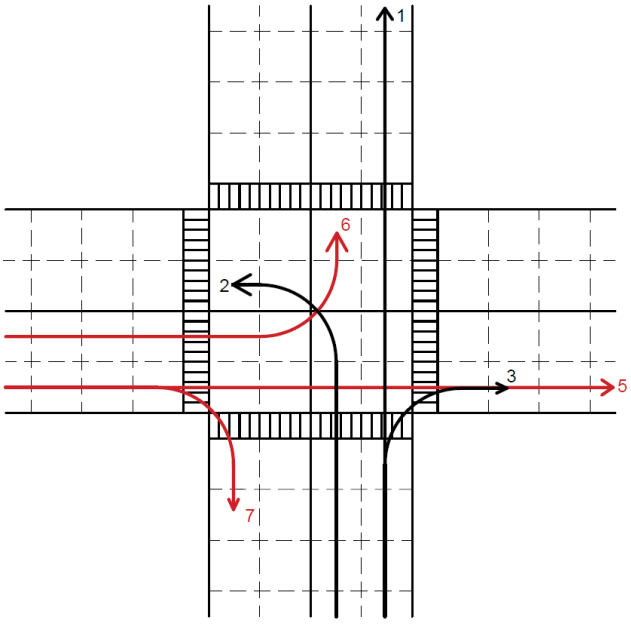
Vytvoření křižovatky se děje za pomocí prostorové a časové diskretizace, což je náhrada kontinuálního (spojitého) systému na systém bodový (diskrétní). Průjezd křižovatkou lze považovat za problém diskretizovaného přidělování zdrojů a optimalizace, kde jsou časové úseky a geografický prostor diskretizovány a přidělovány účastníkům silničního provozu s cílem maximalizovat, případně minimalizovat určitý parametr řízení světelné křižovatky [1]. Příklad jednoduché křižovatky diskretizované do dlaždic je na obrázku 2. Obecně platí, že čím hustší síť (například menší dlaždice), tím přesnější model, avšak to sebou nese vyšší složitost pro návrh algoritmu.



Obr. 2: Diskretizovaná křižovatka [1]

3.1.2 Modelace trajektorie

Při průjezdu křižovatkou vozidla obecně sledují určité vzorce pro její projíždění v závislosti na pravidlech předem definované trasy. V případě, že je trajektorie cesty sledována, je snadné identifikovat trajektorie, které se vzájemně kříží (jsou v konfliktu) a které ne. Nekonfliktní trajektorie potom tvoří tzv. bezpečný vzor v jehož rámci jsou vozidla schopna projet dopravní křižovatkou [1]. Cílem je plánovat trasy pro různé pohybující se objekty bez konfliktů, tedy minimalizovat počet cest, které se překrývají. Na obrázku 3 jsou znázorněny trajektorie. Sady (1, 2), (2, 3), (2, 4) atd. jsou bezpečné trasy, oproti (2, 5), (2, 6) atd., které jsou nebezpečné. Modelování podle plánování trajektorie bylo široce používáno pro řízení a koordinaci leteckého provozu [10] a pro zamezování srážkám vozidel [11].



Obr. 3: Vymodelovaná trajektorie [1]

3.1.3 Modelace oblasti kolize

Potenciální oblast kolize lze předvídat kombinací dvou zmíněných metod, tedy za pomocí diskretizace a vymodelované trajektorie [1]. V tomto případě je třeba vzít v potaz jen potenciální kolizní oblast. Tím se docílí snížení složitosti časových buněk s rezervací prostoru ve vymodelované křižovatce.

3.2 Optimalizace a její metody

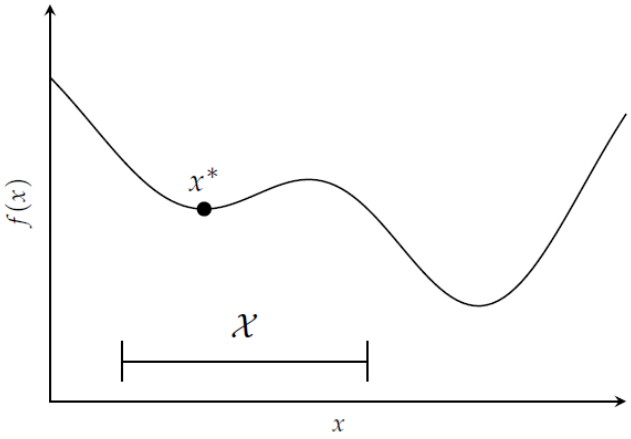
Základní optimalizační problém je minimalizovat (nebo maximalizovat) funkci:

$$f(x) \tag{3.1}$$

kde x ∈ X je bod návrhu, který může být reprezentován jako vektor hodnot odpovídající různým návrhovým proměnným. Pro n – rozměrný prostor je tento bod:

$$[x_1, x_2, \dots, x_n]. \tag{3.2}$$

Prvky ve vektoru 3.2 lze upravit tak, aby byla funkce minimalizována. Jakákoli hodnota x ze všech bodů v proveditelné množině X, která minimalizuje účelovou funkci, se nazývá řešení a je označeno jako x* [12]. Příklad minimalizační funkce je na obrázku 4, kde x* je lokální minimum.

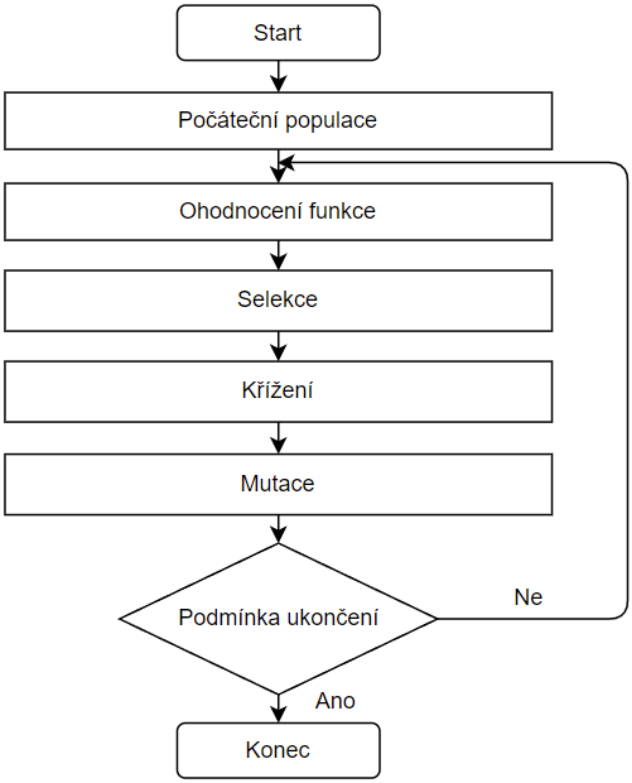


Obr. 4 Jednorozměrný optimalizační problém [12]

Na minimalizaci, případně maximalizaci, určitého problému u světelné křižovatky existuje nepřeberné množství přístupů. Jsou to například různé matematické přístupy, metody založené na pravidlech, strojové učení, genetické algoritmy, dále dynamické programování, přístupy založené na simulaci, posilované učení, neuronové sítě nebo Petriho sítě [4]. Používají se i různé kombinace těchto algoritmů.

3.2.1 Genetický algoritmus

Genetický algoritmus je široce používaná metoda v návrhu časování signálu. Inspirován je přirozenými procesy v přírodě a spadá do evolučních algoritmů. Je založen na Darwinově teorii o vývoji druhů. Jeho úkolem je z počáteční populace různým křížením, mutací a selekcí dostat ideálního jedince [13] (viz obrázek 5).



Obr. 5: Genetický algoritmus [13]

Jedinec je v populaci reprezentován jako chromozom. Ten může být představován jako binární řetězec různé délky nebo jako seznam skutečných hodnot. Použitím binárního chromozomu se lépe aplikuje křížení a mutace, avšak je někdy složitější jeho dekódování [12].

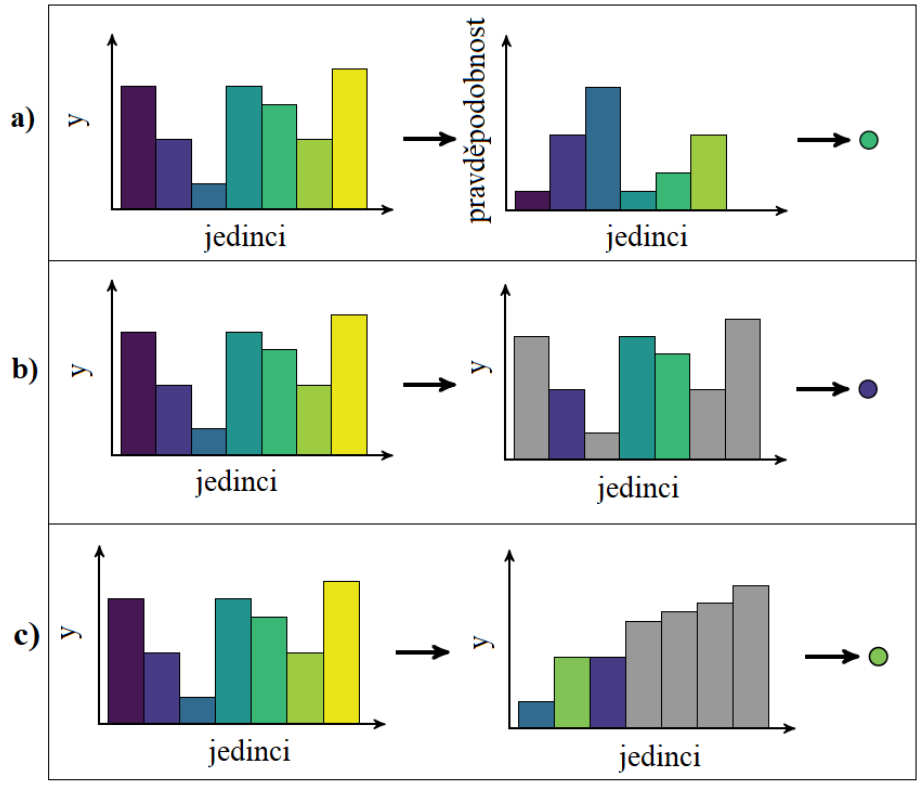
Genetický algoritmus začíná náhodně vygenerovanou populací. Tím, že tvorba populace je založena na náhodě, je jeho nevýhodou neumožnění velké kontroly nad populací [13].

Selekce

Selekce je výběr chromozomů použitých jako rodiče pro další generaci. Existuje několik přístupů, jak selekci provádět [12]:

- ruletová selekce – výběr je náhodný, přičemž čím kvalitnější jedinec, tím vyšší pravděpodobnost, že bude vybrán. Obrázek 6a znázorňuje tento typ selekce. Zde výška sloupce udává hodnotu objektivní funkce (vlevo) a pravděpodobnost, že bude vybrán (vpravo). Každá barva znázorňuje jedince v populaci.
- Turnajová selekce – do nové populace se vkládají nejlepší jedinci z náhodně vybraných n-tic. To je zobrazeno na obrázku 6b, kde sloupce udávají hodnotu objektivní funkce a každá barva odpovídá jedinci z populace.

- Pořadová selekce – pravděpodobnost výběru souvisí s umístěním jedince v posloupnosti, kde jsou chromozomy seřazeny podle kvality. Viz obrázek 6c, ve kterém stejně jako v předchozích případech sloupce označují hodnotu účelové funkce a barva ukazuje, jakému jedinci z populace odpovídá.
- Elitní forma turnajové selekce – vybrán je nejlepší jedinec z populace a dále náhodně vybraní jedinci.



Obr. 6: Vybrané metody selekce genetického algoritmu [12]

Křížení

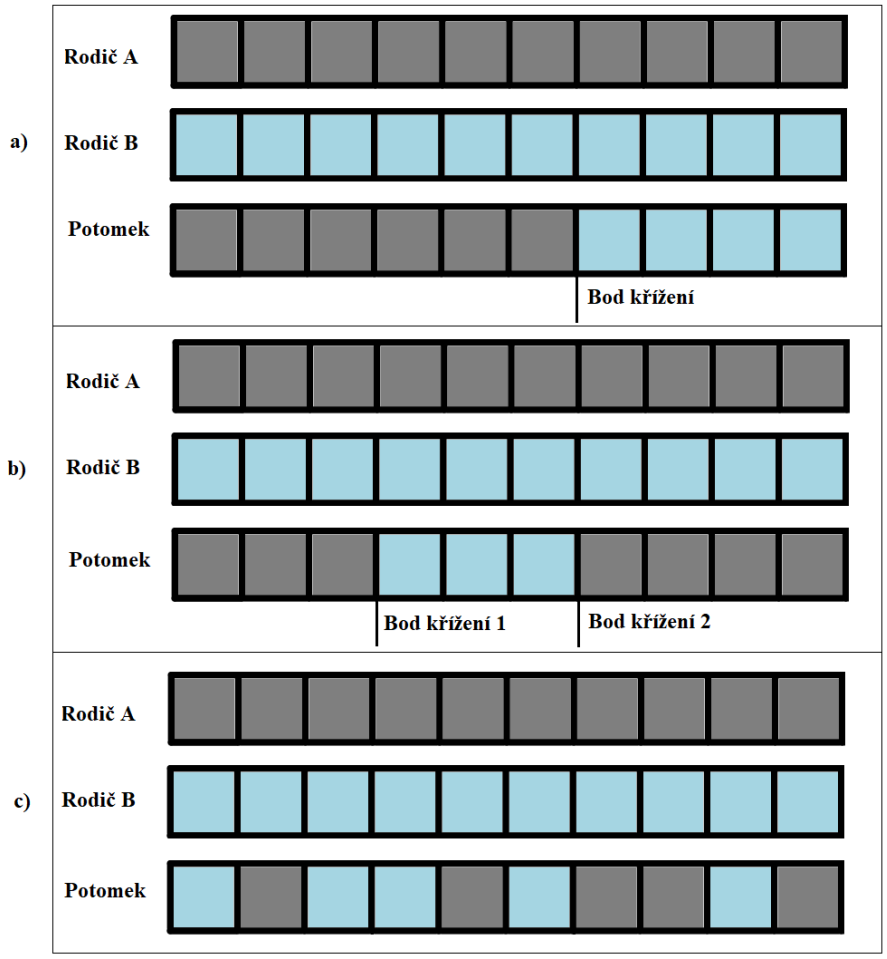
Spojení chromozomů rodičů a dětí se nazývá křížení. Existuje několik druhů [12]:

- jednobodové křížení – je určen náhodný bod křížení, kde dochází k přechodu (obrázek 7a).
- Dvoubodové křížení (obrázek 7b) – použity dva náhodné body křížení,
- Jednotné křížení – založen na pravděpodobnosti (obrázek 7c)

Zmíněné metody křížení fungují pro chromozomy s informací ve formě bitu, ale také pro chromozomy s reálnou hodnotou. Je možné však definovat další metodu křížení, která interpoluje mezi reálnými hodnotami:

$$x \leftarrow (1 - \lambda)x_a + \lambda x_b, \qquad (3.3)$$

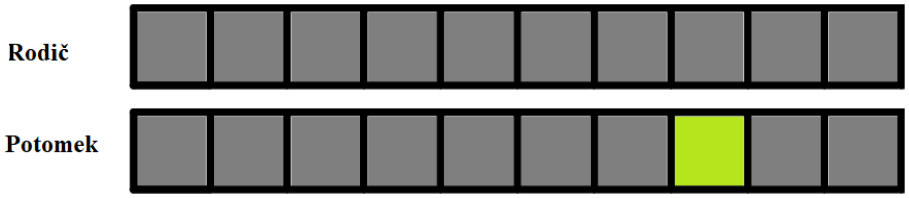
kde: x_a, x_b …rodiče populace,
x …potomek,
λ …konstanta obvykle nastavená na hodnotu 0,5.



Obr. 7: Metody křížení genetického algoritmu [14]

Mutace

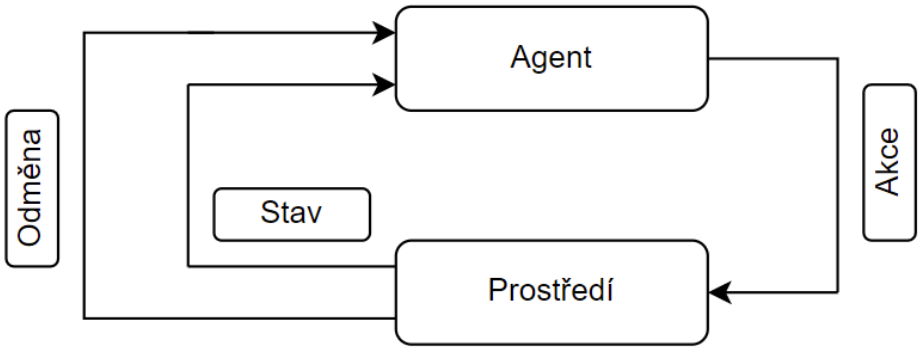
Pro získání nových vlastností, které se neobjevily u křížení slouží mutace [12, 13], která umožňuje objevení těchto dosud nezískaných vlastností (obrázek 8). Tím se docílí prozkoumání větší části stavového prostoru. Pokud by se mutace u algoritmu nevyskytovala hrozilo by tzv. přesycení populace. Pravděpodobnost, že dojde u chromozomu k mutaci bývá nízká. Probíhá tak, že se vezme náhodný bit, který se převrátí ze svého původního stavu.



Obr. 8: Mutace genetického algoritmu [14]

3.2.2 Q – learning

Pro optimalizace světelné křižovatky se často používá Q – learning [15]. Patří do skupiny algoritmů posilovaného učení, což je strategie, která pracuje s autonomním agentem na základě interakce s prostředím (viz obrázek 9). Prostředí informuje agenta o odměně za provedenou akci.



Obr. 9: Interakce agenta s prostředím [16]

Q-učení funguje tak, že postupně zlepšuje svá hodnocení kvality a konkrétních akcí v jednotlivých stavech. Cílem agenta je maximalizovat odměnu, tedy nalezení nejlepšího postupu. Uvažujeme-li nějaký konečný diskrétní svět, tak za každý krok dostává agent pravděpodobnostní odměnu [17]. Agent běží v cyklech a v každé iteraci pozoruje stav prostředí. Na základě pozorování provede akci, čímž převede prostředí do nového stavu. Za provedení akce dostane již zmíněnou odměnu. Iterace běží, dokud není splněna některá podmínka. V optimalizaci křižovatky tento algoritmus vyžaduje velké množství stavů a akcí.

3.2.3 Petriho sítě

K modelování a optimalizaci dopravních stavů se dále používá metoda Petriho sítí. Například v [18] byly použity k vyjádření problému koordinace semaforů s cílem zlepšit výkon tranzitních vozidel.

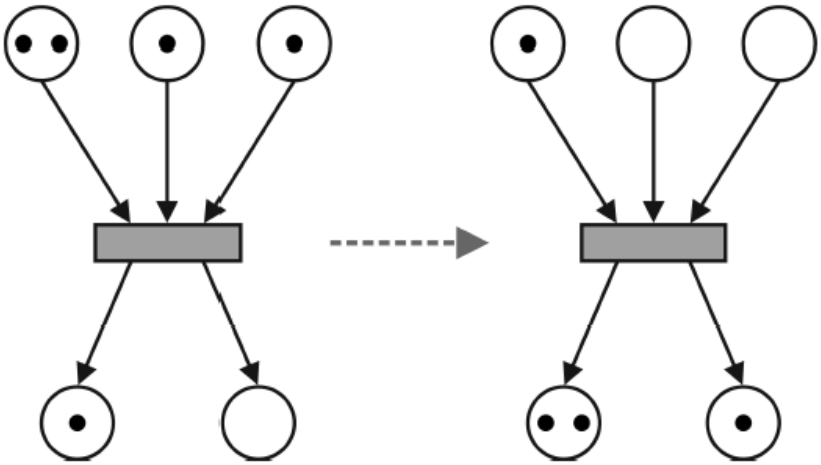
Petriho síť je matematická reprezentace diskrétních systémů. Jedná se o jejich grafický popis a analýzu. Těchto sítí existuje několik typů, například [19]:

- C/E (Condition/Event),
- TPN časované (Timed),
- HPN Hierarchické (Hierarchical) apod.

Každá síť se skládá z:

- places (místa) – je v nich uložena stavová informace ve formě tokenů (značek) a jsou znázorněny kružnicemi,
- transitions (přechody) – vyjádření možných změn stavů, znázorněny obdélníky,
- arcs (hrany) – určení logické vazby, znázorněny šipkami.

Petriho síť je obecně definována jako n-tice, kde je obsažena množina míst, přechodů váhových funkcí, směrových oblouků, signálů [20] a dalších množin podle typu použité sítě. Jednotlivé přechody mezi stavy se uskutečňují následovně. Pokud je v místě značka, znamená to, že je splněna podmínka a přechod je umožněn. Při přechodu dojde ke změně, a to odebrání značky ze vstupního místa a přidání značky na místo výstupní (viz obrázek 10).



Obr. 10: Přechod u Petriho sítě [19]

3.2.4 Numerické metody

K nalezení optimálního řešení při minimalizaci nebo maximalizaci hledaného parametru pro vyhodnocování světelné křižovatky slouží i různé numerické metody. Tyto úlohy se zabývají výběrem nejlepších řešení z dané množiny možných řešení [21]. Těchto úloh je velké množství a v této práci budou použity dvě následující.

Powellova metoda

Přesněji Powellova metoda konjugovaného směru, je algoritmus navržený pro nalezení lokálního minima funkce. Je vyvinuta pro konvexní kvadratický problém, ale lze ji úspěšně aplikovat na nekvadratické problémy [22]. Funkce musí mít skutečnou hodnotu a pevný počet vstupů. Algoritmus postupně prochází sadou počátečních směrů (vektorů), jež představují normály zarovnané ke každé ose a minimalizuje funkci obousměrným prohledáváním podél každého vyhledávacího vektoru [23]. Při prohledávání pokaždé aktualizuje návrhový bod [12]:

$$x^{(i+1)} \leftarrow \text{řádkové prohledávání}\left(f, x^i, u^i\right) \text{ pro všechna } i \text{ v } \{1, \dots, n\} \qquad (3.4)$$

kde $u^{(1)}, \dots, u^{(n)}$ je směr hledání, což jsou z počátku souřadnicové základní vektory $u^{(i)}$ pro všechna i. Počáteční hodnota je $x^{(1)}$. Následně se všechny směry posunou dolů o jeden index a nejstarší směr prohledávání ($u^{(1)}$) je vypuštěn:

$$u^{(i)} \leftarrow u^{(i+1)} \text{pro všechna } i \text{ v } \{1, \dots, n-1\} \quad (3.5)$$

Poslední směr hledání je nahrazen směrem od $x^{(1)}$ do $x^{(n+1)}$, což je celkový směr postupu za poslední cyklus:

$$u^{(n)} \leftarrow x^{(n+1)} - x^{(1)} \quad (3.6)$$

a další řádkové vyhledávání se provádí v novém směru k získání nového $x^{(1)}$. Tento proces se opakuje až do konvergence.

Výběr směrů se často liší, může být však následující [24]:

1) začne se s počátečním odhadem a minimalizuje se ve směru základního vektoru $x^{(1)}$.
2) Z výsledného bodu se minimalizuje ve směru druhého vektoru $x^{(2)}$.
3) Pokračuje se tímto způsobem až do $x^{(n)}$.
4) Dále postupuje cyklicky přes $x^{(1)}$ až $x^{(n)}$, dokud již nedojde k žádné významné změně.

Tato metoda nachází úspěšně minimum, avšak je velmi neefektivní, jelikož výpočet trvá velmi dlouho [23].

Nelderova-Meadova metoda

Též známá jako metoda simplex. Slouží ke hledá minima účelové funkce porovnáváním jejích hodnot v určitých bodech prostoru. Těmito body jsou myšleny vrcholy simplexu. V rozměrech N je simplex geometrický útvar skládající se z N + 1 bodů a spojovacích čar, rovin. Například ve dvou rozměrech je simplex trojúhelník a ve třech rozměrech je to čtyřstěn [21].

Pomocí počátečního bodu se provádí kroky při každé iteraci, aby bylo nalezeno (lokální) minimum. Nechť jsou simplexové body $x^{(1)}, \dots, x^{(n+1)}$ s vrcholem x_s, který má nejvyšší hodnotu. Dále vrchol x_l s nejnižší hodnotou funkce. Průměr všech bodů kromě nejvyššího vrcholu x_h nechť je $\bar{x}$. Kroky, které může tento algoritmus podniknout, se vyhodnocují ve čtyř různých operacích. [12, 24]:

- Reflexe – algoritmus vezme nejvyšší bod z $x^{(n+1)}$ bodů, které definují simplex, a zrcadlí jej na opačnou stranu simplexu, tak aby byl zachová jeho objem (obrázek 11a):

$$x_r = \bar{x} + \alpha(\bar{x} - x_h), \quad (3.7)$$

  kde α>0 a obvykle je nastaveno na 1.
- Expanze – simplex se rozšiřuje ještě dále (obrázek 11b):

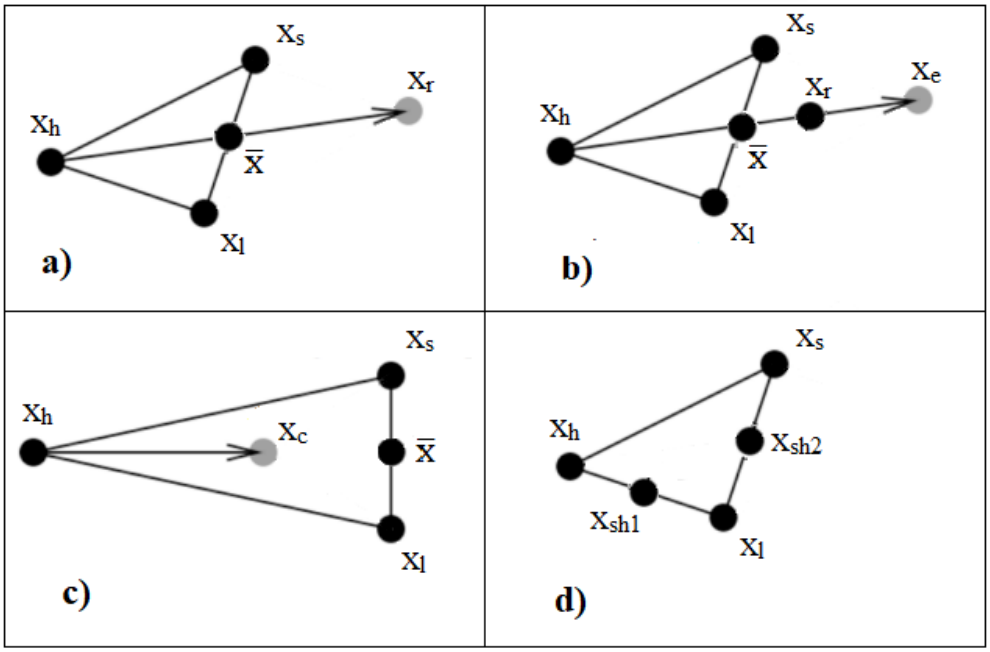
$$x_e = \bar{x} + \beta(x_r - \bar{x}), \quad (3.8)$$

  zde β>max (1, α) a obvykle se udává na 2.
- Kontrakce – když se simplex přiblíží minimu, sám se stáhne v transponovaném směru (obrázek 11c).

$$x_c = \bar{x} + \gamma(x_h - \bar{x}), \qquad (3.9)$$

kde $\gamma \in (0, 1)$ a typicky bývá nastaveno na 0,5.

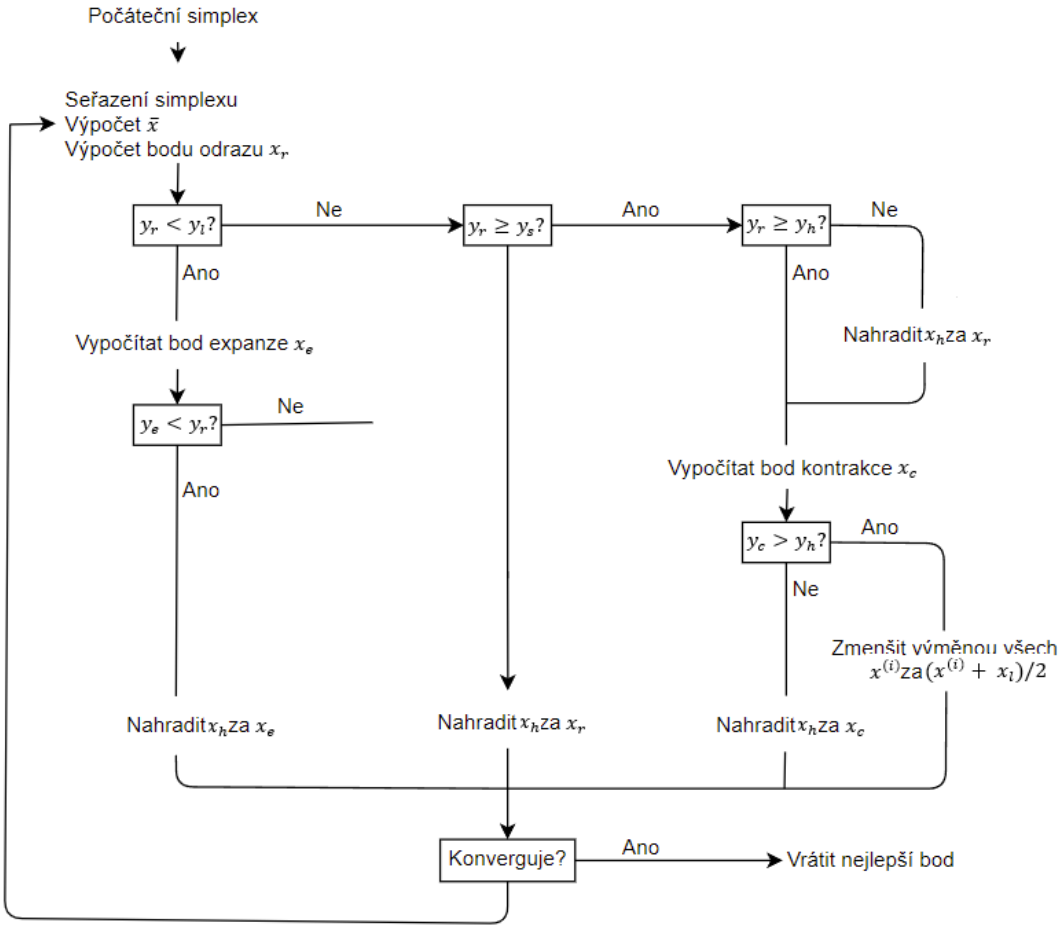
- Srážení – všechny body jsou přesunuty směrem k nejlepšímu, obvykle je to polovina separační vzdálenosti. Na obrázku 11d to jsou body x_{sh1} a x_{sh2}.



Obr. 11: Možné pohyby Nelderovy-Meadovy metody [24]

Výše zmíněný postup je znázorněn na vývojové diagramu (obrázek 12), kde pro jakýkoliv návrhový bod $x_{Ø}$ nechť $y_{Ø} = f(x_{Ø})$ je jeho účelová funkce.

Nelderova-Meadova metoda je heuristická a vhodná pro minimalizaci funkcí s menším počtem proměnných (n <10) [21]. Kritériem konvergence je v tom, že bere v úvahu odchylky ve funkčních hodnotách spíše něž změny bodů v návrhovém prostoru. Porovnává směrodatnou odchylku vzorku $y^{(1)}, \ldots, y^{(n+1)}$ na toleranci Q. Tato hodnota je vysoká pro simplex nad vysoce zakřivenou oblastí a nízká pro simplex nad plochou oblastí. Když je tedy oblast vysoce zakřivená, značí to, že je možná další optimalizace [12]. Nevýhodou této metody je pomalost výpočtu, zejména v oblasti minima.



Obr. 12: Vývojový diagram Nelderova-Meadova algoritmu [12]

3.3 Simulační nástroje

Již bylo zmíněno v úvodu kapitoly, že existuje velké množství simulačních nástrojů, sloužících k vymodelování přesné dopravní sítě a její následné simulaci. Tyto nástroje, především jimi vytvořené dopravní modely, se jeví jako velmi užitečné při analýze a optimalizaci dynamického chování dopravní sítě.

Simulaci lze definovat, jako imitaci reálných systémů nebo procesů pro bezproblémové získávání informací prostřednictvím modelů dopravních toků [2]. Tyto modely poté napomáhají při popisu fyzického chování dopravní sítě. Jejich využití je klíčové pro komplexní průzkum městského dopravního systému.

Simulační nástroje lze rozdělit na mikroskopické a makroskopické [2]. Makroskopické berou v úvahu proudění vozidel jako celek a hodí se spíše pro velkou dopravní síť. Zatímco mikroskopické uvažují individuální chování řidiče spolu s interakcí s ostatními účastníky silničního provozu, takže je ideální pro analýzu konkrétní křižovatky. Základní přehled mikro simulačních nástrojů je v tabulce 1.

Tab. 1: Přehled nečastějších simulačních nástrojů [2]

	CORSIM	**MATSim**	**AIMSUN**	**Paramics**	**VISSIM**	**SUMO**
Volně přístupný	Ne	Ano	Ne	Ne	Ne	Ano
Systém	Diskrétní	Spojitý	Spojitý	Diskrétní	Spojitý	Spojitý
Vizualizace	2D, 3D	2D	2D, 3D	2D, 3D	2D, 3D	2D, 3D
Chodci	Ano	Ne	Ano	Ano	Ano	Ano
MHD	Ne	Ne	Ano	Ano	Ano	Ano
Oblast působnosti	Město/kraj	Kraj/stát	Město/kraj	Město/kraj	Město/kraj	Město/kraj
Výstup	xml, csv	text	Grafy	html, xml, csv	xml	xml
Import map	N/A	Ano	Ano	N/A	Ano	Ano
Programovací jazyk	N/A	N/A	Python, C++	N/A	C++, VB, Matlab, Python,..	C++, VB, Matlab, Python,..

Z tabulky 1 je patrné, že vlastnosti simulačních nástrojů se různí. Většina má oblast působnosti omezenou na město či kraj. To se hodí především na modelování podrobnějších částí dopravní infrastruktury, jako jsou křižovatky apod. Z toho důvodu má převážná část těchto simulačních programů možnost zobrazení chodců a městské hromadné dopravy. Ze zmíněných má jen program MATSim velkou oblast působnosti, kdy umožňuje spustit simulaci i nad konkrétním státem. Jeho nevýhodou je malá podrobnost, avšak v některých případech to nemusí být žádoucí.

Některé nástroje umožňují i import map. Po vybrání určité oblasti výzkumu na mapě, se automaticky vytvoří síť, kterou většinou stačí jen drobně upravit. Tím se ušetří velké množství času, který by se strávil na vytvářením modelu. Dále stojí za zmínku podpora různých programovacích jazyků, což činí simulační programy mnohem přístupnější.

Dle výše uvedeného proběhl výběr nástroje pro potřeby simulace světelné křižovatky řešené v této diplomové práci. Byl zvolen program SUMO, a to z následujících důvodů. Je volně přístupný, takže se nemusela řešit žádná licence. Dále umožňuje i simulací chodců, což je pro řešenou světelnou křižovatku žádoucí. Jeho oblast působnosti, která se zaměřuje na město nebo kraj je taky vyhovující. V neposlední řadě umožňuje spouštět a nastavovat simulaci v programovacím jazyku Python, což je jazyk, který byl zvolen na softwarovou implementaci problému v této závěrečné práci.

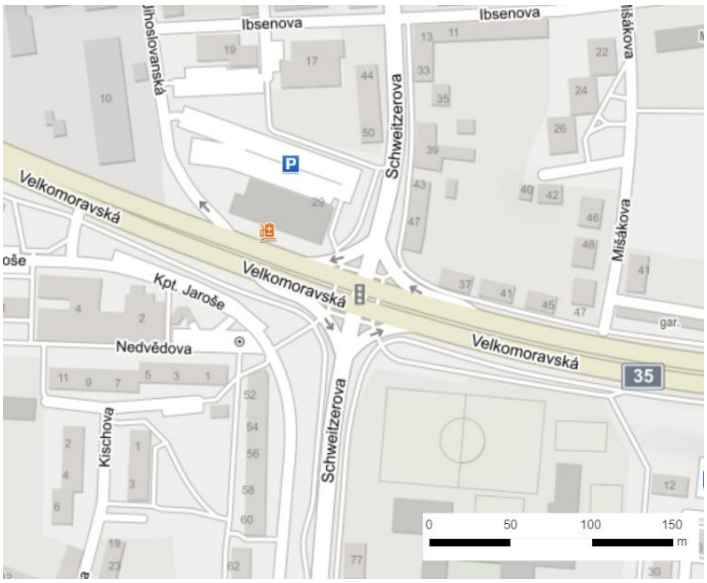
4 SOFTWAROVÁ IMPLEMENTACE PROBLÉMU

Pro řešení optimalizace světelné křižovatky bylo třeba vybrat konkrétní křižovatku a pro ni vytvořit simulační model. Byly sestaveny dva modely, jeden pro běžný provoz a druhý pro dopravní špičku. Následovalo nastavení fází semaforů a jejich časování. Poté se přešlo ke konkrétním metodám optimalizace. Hledalo se ideální nastavení časování semaforů tak, aby parametr „celkové zpoždění odjezdu všech vozidel“ měl co nejmenší hodnotu.

Křižovatka pro optimalizaci byla zvolena ve městě Olomouc na ulicích Velkomoravská a Schweitzerova. Jako simulační nástroj byl vybrán program SUMO a programovací jazyk Python, který je s tímto simulačním nástrojem kompatibilní díky knihovnám sumolib a traci. Hledání minima potom probíhalo za pomocí genetického algoritmu, Powellovy a Nelderovy-Meadovy metody. Nakonec se porovnaly jednotlivé metody mezi sebou pro oba modely, a ještě se srovnaly s reálnou situací a se základním nastavením časování, které automaticky nabídl program SUMO při vytváření dopravní sítě. Vše bude podrobněji popsáno v následujících kapitolách.

4.1 Křižovatka k řešení

Byla vybrána křižovatka ve městě Olomouc (obrázek 13), která kříží ulice Velkomoravská a Schweitzerova. Ta odděluje sídliště Povel a centrum města. Z toho plyne, že je křižovatka velmi frekventovaná všemi účastníky silničního provozu, tedy vozidly i chodci a nalezení optimálního řešení pro její přechod je velmi žádoucí.



Obr. 13: Optimalizovaná křižovatka [25]

4.2 Použité technologie

Již bylo řečeno, že pro simulaci světelné křižovatky byl použit program SUMO s využitím programovacího jazyka Python. Tento jazyk a jeho knihovny byl použit i pro implementaci optimalizačních metod.

4.2.1 SUMO

„Simulation of Urban MObility" [26] tedy simulace městské mobility (SUMO), je volně přístupný software pro simulaci provozu a je navržený pro práci s velkými sítěmi. Jeho vývoj začal roce 2000 z důvodu nedostatku nástrojů na trhu se schopností implementovat a vyhodnocovat vlastní algoritmy.

Je kladen důraz na rychlost simulace, proto se nástroj dá spustit jen z příkazové řádky. Navíc to i simulační algoritmus zjednodušuje. Jednotlivé body, hrany, trasy aj., se dají ukládat jako samostatné xml soubory, ze kterých se poté vytvoří konečná síť. Ta se již nemusí více generovat a simulační algoritmus se vždy aplikuje na tuto síť.
Kromě vozidel umožňuje i simulaci chodců. Má velkou sadu nástrojů pro tvorbu nejrůznějších scénářů, jako například různorodé nastavení simulace, import vlastní sítě, vytvoření dopravní sítě přímo z místa na mapě a mnoho dalšího.

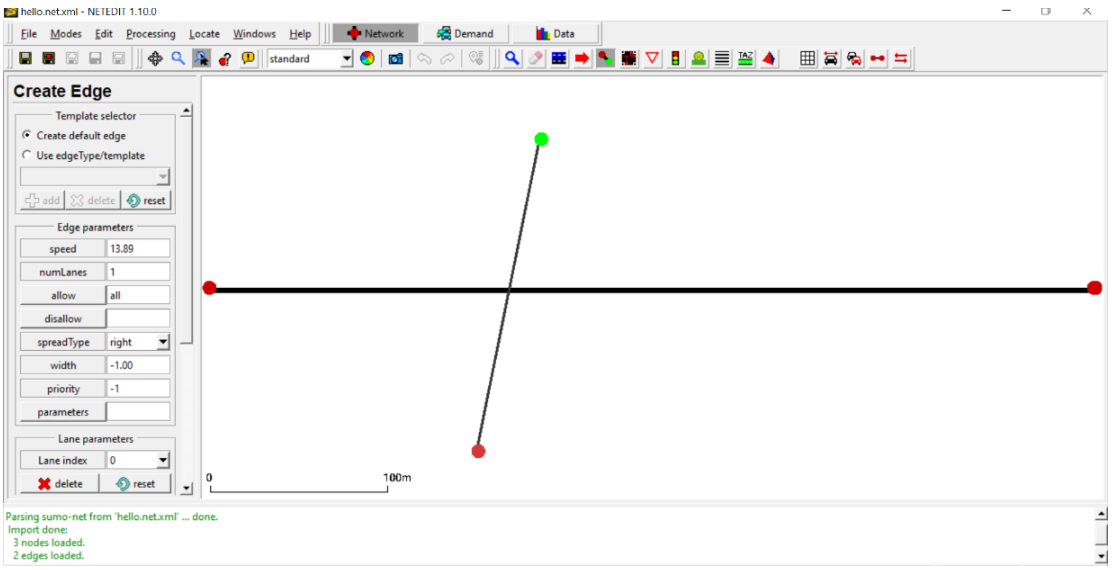
Vytváření dopravní sítě
Jedná se o vytvoření souboru ve formátu xml, ve kterém jsou obsaženy všechny informace ohledně dopravní sítě [26]. Konkrétní síť obsahuje tyto informace:

- hrany neboli silnice – jejich tvar, poloha a omezení rychlosti,
- logika semaforů,
- uzly neboli křižovatky – včetně jejich regulace přednosti v jízdě,
- a další parametry v závislosti na definovaném vstupu.

Tento soubor není doporučeno vytvářet ani editovat ručně. Jeho vygenerování se děje následujícími způsoby:

- příkaz *netconvert* – nejprve je potřeba vytvořit xml soubory pro uzly, hrany, případně další soubory potřebně pro vymodelování konkrétní dopravní situace. Příkaz potom vypadá následovně:
  - *netconvert --node-files=test.nod.xml --edge-files=test.edg.xml \ -- output-file=jedn.net.xml,*

  kde *node-files* definuje uzly a *edge-files* hrany.
- Vizuální editor (*netedit)* – vybudování dopravní sítě z grafického rozhraní (ukázka na obrázku 14).
- Příkaz *netgenerate* – generuje abstraktní silniční sítě.
- OSMWebWizard – nástroj, pro vytvoření sítě z konkrétního místa na mapě.



Obr. 14: Netedit – nástroj na vytváření sítě

Spouštění simulace dopravní sítě

Již bylo zmíněno, že se simulace dá spustit z příkazové řádky. Existuje ovšem také možnost ji spustit v grafickém rozhraní. Pro potřeby optimalizace je vhodnější první varianta, jelikož běží velké množství simulací a není potřeba každou z nich sledovat. Ovšem k zobrazení výsledného optimalizovaného řešení se grafické rozhraní jeví jako příhodné.

Ještě před samotným spuštěním se musí vytvořit účastníci silničního provozu a jejich trasy. To je možné ve zmíněném editoru *netedit* nebo vlastnoručně vytvořeným souborem ve formátu xml, ve kterém jsou tyto informace uloženy. Příklad jeho názvu je *test.rou.xml*.

Pro úspěšné spuštění je zapotřebí mít v datech konfigurační soubor s příponou *sumocfg*. Který vezme vstupní informace (vytvořenou síť a trasy účastníků provozu) a podle příkazu spustí simulaci buď v příkazovém řádku, nebo v grafickém rozhraní. Příkazy ke spuštění jsou:

- *sumo -c test.sumocfg* – příkazový řádek,
- *sumo-gui -c test.sumocfg* – grafické rozhraní,

kde *test.sumocfg* je zmiňovaný konfigurační soubor.

4.2.2 Python

Jedná se o vysokoúrovňový programovací jazyk s jednoduchou syntaxí a dynamickým psaním. Je objektově orientovaný a nabízí dynamickou kontrolu různých datových typů, nejrůznější moduly a třídy. Je dostupný pro většinu běžných platforem, jako je Unix, MS Windows, macOS, Android. Python se používá k vývoji různých aplikací, jako jsou webové aplikace, aplikace pro vývoj softwaru, síťové programování, hry a 3D aplikace a další aplikace

Existuje asi 200 základních modulů implementovaných jíž při instalaci Pythonu. Dále je dostupné velké množství knihovních modulů ke stažení, které bývají většinou

volně přístupné [27]. Pro optimalizaci světelné křižovatky byly použity následující knihovny:

NumPy
Určená pro vědecké výpočty, která poskytuje vícerozměrný objekt pole, různé odvozené objekty (např. matice) a řadu nástrojů pro rychlé operace s poli [28] včetně matematických a logických manipulací. Dále nabízí diskrétní Fourierovy transformace, základní lineární algebru, statistické operace a mnoho dalšího.

Matplotlib
Knihovna sloužící primárně pro vytváření statických, animovaných i interaktivních grafů [29]. Umožňuje:

- Vytvářet rozličné grafy,
- interaktivní grafy – ty lze přibližovat, posouvat a aktualizovat,
- export do rozlišných formátů souborů (png, jpg, pdf, atd.).

BeautifulSoap
Nástroj na získávání dat souborů ve formátu xml a html. S dokumentem pracuje jako s objektem, takže se tak k němu může i přistupovat [30]. To umožňuje rychlou práci s daty.

ElementTree
Slouží k vytváření a čtení souborů ve formátu xml. Celý xml dokument představuje jako strom prvků [31]. Na této úrovni provádí interakce s celým dokumentem.

Sumolib
Sada modulů pro práci se sítěmi SUMO, výstupem simulace a dalšími artefakty simulace [26].

Traci
Zpřístupněním běžící simulace silničního provozu v programu SUMO [26]. Umožňuje získávat hodnoty simulovaných objektů a manipulovat.

Scipy
Tato knihovna je sbírka matematických algoritmů a praktických funkcí postavená na rozšíření NumPy. Přidává významnou sílu interaktivní relaci Pythonu tím, že poskytuje uživateli příkazy a třídy na vysoké úrovni pro manipulaci a vizualizaci dat [32]. Umožňuje vše od paralelního programování, až po webové a databázové podprogramy. Pro potřeby diplomové práce byla použita, protože má v sobě implementované metody optimalizace.

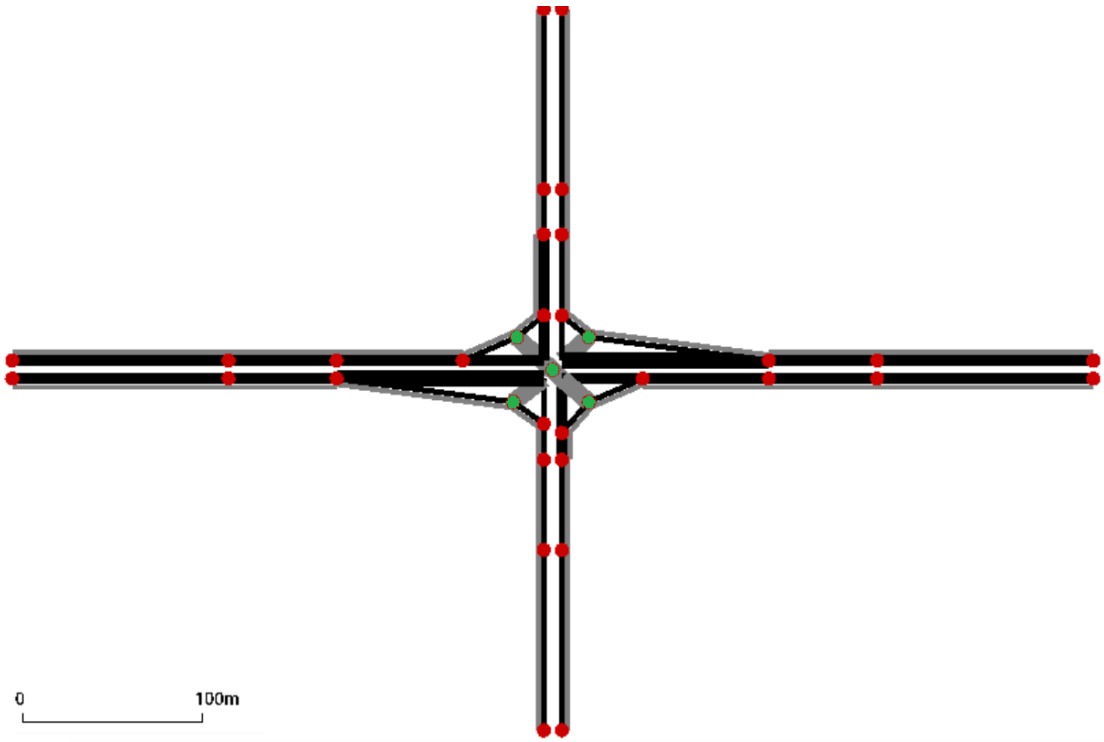
4.3 Vytvoření modelu

K vytvoření modelu bylo třeba získat dopravní síť. K tomu bylo třeba nadefinovat jednotlivé uzly, hrany, spojení a trasy. Pro tento účel byl vytvořen skript v Pythonu, který pro každý zmíněný parametr vytvořil xml soubor. Poté již bylo možné, za pomocí konfiguračních souborů z programu SUMO, využít nástroj, který vygeneroval světelnou křižovatku jako simulační síť opět jako xml soubor. V něm se již nacházelo základní nastavení fází, které bylo dále upraveno. Vše zde zmíněno bude probráno podrobněji níže.

Model se vytvářel dvakrát. Poprvé s hustým provozem a podruhé pro běžný provoz. Oba vytvořené modely jsou v příloze (*\data_to_optimalized*) a nástroj na jejich vytvoření je viz příloha *prepare_cross.py*.

4.3.1 Uzly

Bylo zapotřebí nadefinovat množství uzlů, které spojují jednotlivé cesty, tak aby co nejvíce odpovídaly realitě. Takovýchto bodů se vytvořilo 33, z toho 5 představuje světelné zařízení. Viz obrázek 15, jednotlivé uzly zde jsou jako červená kolečka, semafory jsou zde znázorněny zeleně.

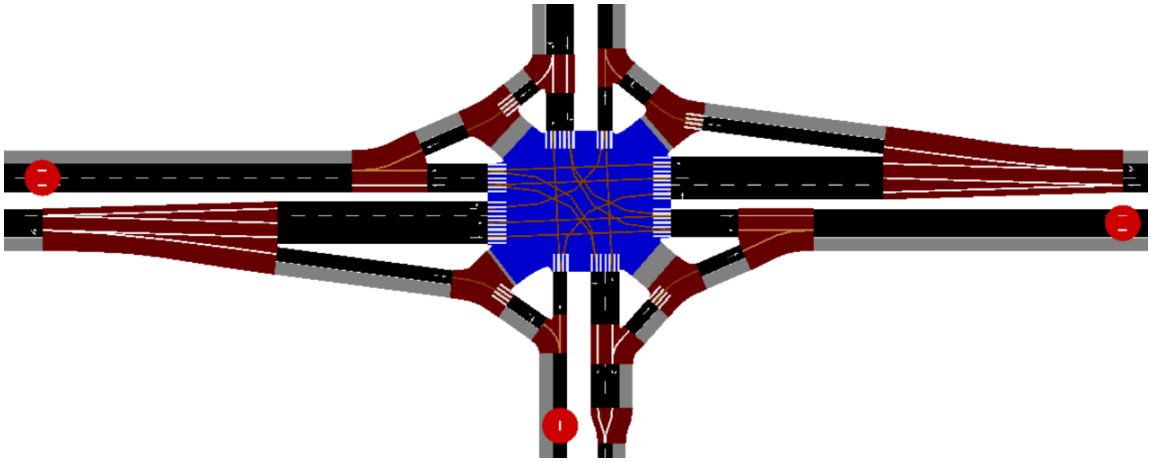


Obr. 15: Nadefinované uzly modelované světelné křižovatky

4.3.2 Hrany a jejich propojení

Jakmile byly nachystány uzly, přišlo na řadu spojení těchto bodů a teda nadefinování jednotlivých hran. Vytvářely se jak dopravní cesty pro vozidla, tak i chodníky a přechody pro chodce. S tím, že pro vozidla se stanovila maximální možná rychlost na této cestě 50 km/h.

Poté přišlo na řadu nastavit možné spojení vzniklých hran. Tedy přesné určení, z jaké cesty a kam má určité vozidlo povoleno vjet (obrázek 16). Základní byly nastaveny automaticky, ale například odbočování doleva se muselo nastavit ručně.



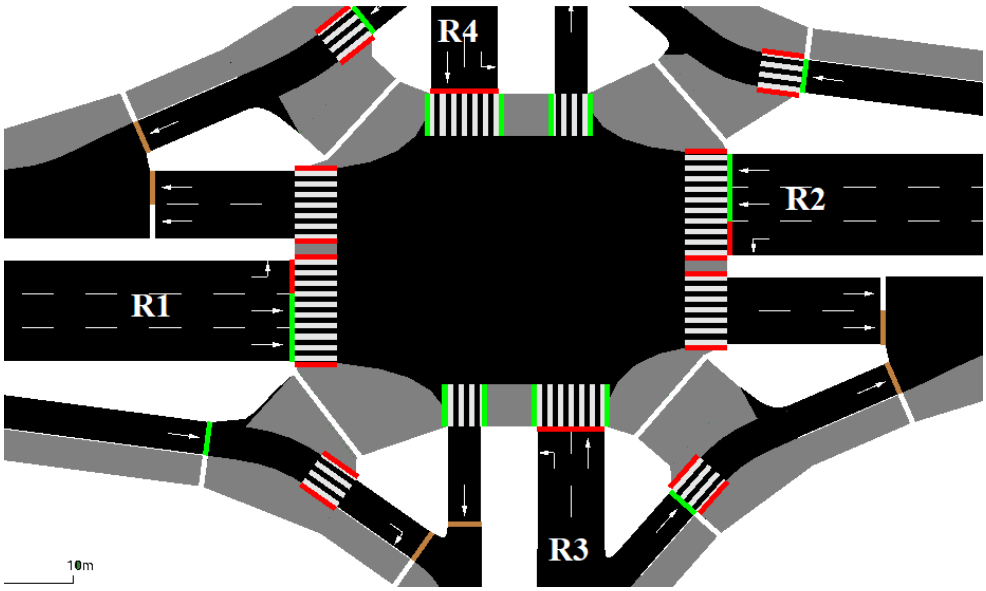
Obr. 16: Propojení jednotlivých hran

4.3.3 Trasy

V neposlední řadě se nastavili různé cesty pro všechny účastníky silničního provozu, kteří se vytvářeli také. Vytváření vozidel a chodců probíhalo na základě pravděpodobnosti, viz tabulka 2 a obrázek 17. Pravděpodobnosti byly získány odhadem.

Tab. 2: Pravděpodobnosti směru jízdy

Trasa	**Pravděpodobnost směru [-]**		
	Vlevo	Doprava	Rovně
R1	0,2	0,3	0,99
R2	0,2	0,2	0,99
R3	0,2	0,2	0,1
R4	0,2	0,2	0,1



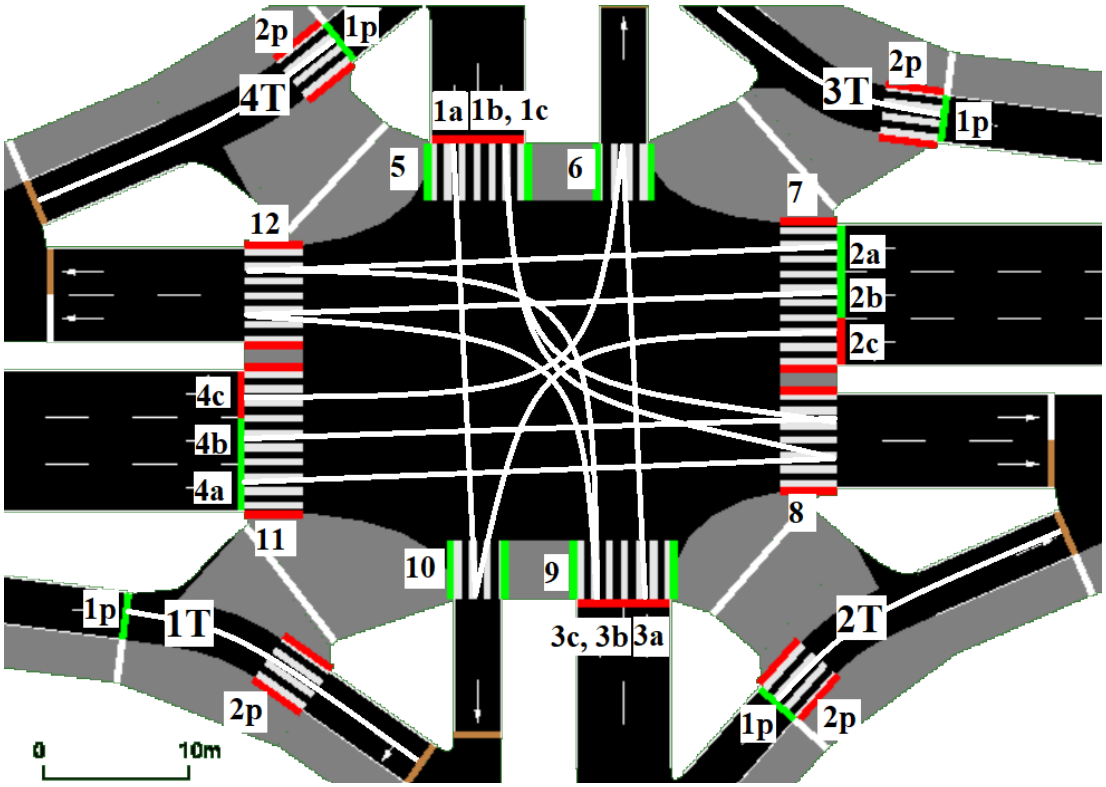
Obr. 17: Trasy vozidel

Jak je patrné z tabulky 2, nejvíce vozidel směřuje rovně po čtyřproudé silnici (ulice Velkomoravská). Hustota vozidel u zbytku směrů již je více méně podobná. Vytvořené dopravní prostředky byly jen jednoho typu, a to obyčejné automobily. Pro to, aby se v dopravní síti ocitli chodci byla pravděpodobnost 0,8. Jejich trasy však byly zcela náhodné.

Tvořili se dva modely, jeden pro dopravní špičku a druhý pro běžný provoz. V případě dopravní špičky se při generování účastníků silničního provozu použil vyšší počet kroků pro iteraci k vytváření vozidel, konkrétně 300. To způsobilo hustší dopravní síť. Naopak pro obyčejný provoz to bylo 50 těchto iterací.

4.3.4 Fáze

Nejdůležitější část vytváření modelu byla příprava jednotlivých fází, jejichž optimalizací se hledala minimální doba celkového zpoždění odjezdu všech vozidel. Na obrázku 14 jsou znázorněny jednotlivé semafory pro všechny povolené směry, jak pro vozidla, tak pro chodce. Na zmíněném obrázku je fáze číslo 1. Ostatní fáze pro hlavní semafor jsou poté v tabulce 3, pro semafor vedlejší, který je na obrázku 18 označen 1T až 2T, to je tabulka 4. Z ní je i patrné, které konkrétní fáze se optimalizovaly. Některé byly pevně dané a to takové, kdy se přecházelo přes žlutou. Doba trvání přepnutí zelená-žlutá-červená trvá 3 sekundy, v opačném pořadí, tedy červená-žlutá-zelená činí 2 sekundy. Tyto časy byly získán z [33].



Obr. 18: První fáze světelné křižovatky

Tab. 3: Fáze světelné křižovatky pro hlavní semafor

Hlavní semafor																					
Číslo fáze	Pro vozidla												Přechod pro chodce								Trvání fáze [s]
	1a	1b	1c	2a	2b	2c	3a	3b	3c	4a	4b	4c	5	6	7	8	9	10	11	12	
1	r	r	r	G	G	r	r	r	r	G	G	r	G	G	r	r	G	G	r	r	Opt.
2	r	r	r	G	G	r	r	r	r	G	G	r	r	r	r	r	r	r	r	r	Opt.
3	r	r	r	y	y	r	r	r	r	y	y	r	r	r	r	r	r	r	r	r	3
4	r	r	r	r	r	y	r	r	r	r	r	r	r	r	r	r	r	r	r	r	2
5	r	r	r	r	r	G	r	r	r	r	r	G	r	r	r	r	r	r	r	r	Opt.
6	r	r	r	r	r	y	r	r	r	r	r	y	r	r	r	r	r	r	r	r	3
7	y	y	y	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	2
8	G	g	g	r	r	r	r	r	r	r	r	r	r	G	G	G	G	r	G	G	Opt.
9	G	g	g	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	Opt.
10	y	y	y	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	3
11	r	r	r	r	r	r	y	y	y	r	r	r	r	r	r	r	r	r	r	r	2
12	r	r	r	r	r	r	G	g	g	r	r	r	G	r	G	G	r	G	G	G	Opt.
13	r	r	r	r	r	r	G	g	g	r	r	r	r	r	r	r	r	r	r	r	Opt.
14	r	r	r	r	r	r	y	y	y	r	r	r	r	r	r	r	r	r	r	r	3
15	r	r	r	y	y	r	r	r	r	y	y	r	r	r	r	r	r	r	r	r	2

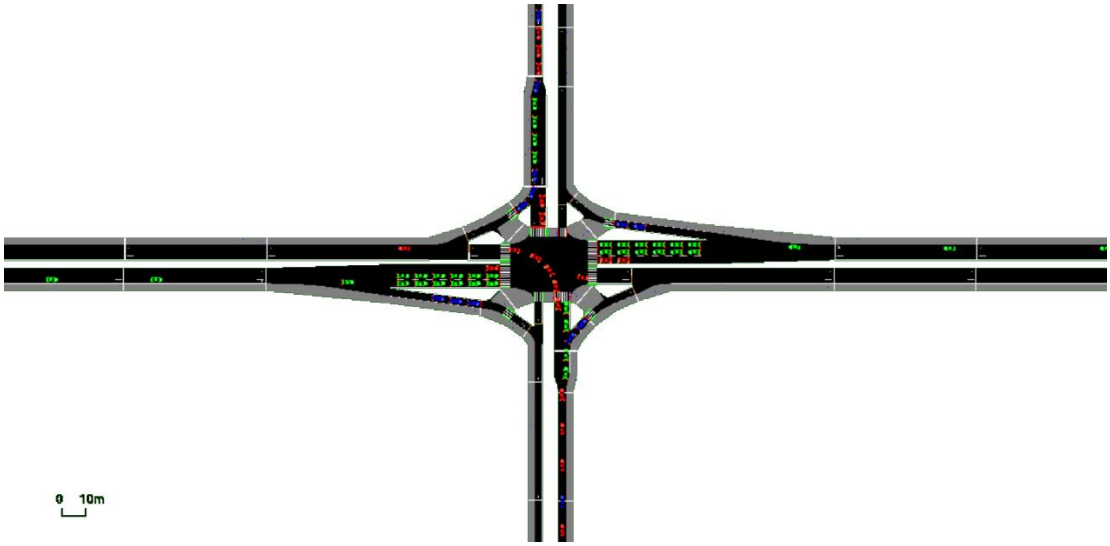
Tab. 4: Fáze světelné křižovatky pro vedlejší semafory

Vedlejší semafory			
Číslo fáze	1p	2p	Trvání fáze [s]
1t	G	r	Opt.
2t	y	r	3
3t	r	G	Opt.
4t	r	r	1
5t	y	r	2

Zde: r… zastavení,
y…zpomalení,
G…volné projetí,
g…projetí, ale vozidlo musí dát přednost v jízdě.

4.3.5 Výsledný model

Nadefinováním výše zmíněných parametrů, tedy uzlů, hran, propojení, tras a fází vznikla výsledná světelná křižovatka (obrázek 19). Pro větší přehlednost byly pro potřeby simulace v prostředí SUMO vozidla barevně rozlišeny. Zelená auta mají udaný směr rovně, modrá odbočují vpravo a červená vlevo.



Obr. 19: Výsledná vymodelovaná světelná křižovatka

Výše bylo zmíněno, že simulace bude probíhat při dopravní špičce a potom při běžném provozu. Množství vygenerovaných účastníků silničního provozu na základě pravděpodobnosti (viz výše tabulka 2) pro obě dvě varianty udává tabulka 5.

Tab. 5: Počet vygenerovaných účastníků silničního provozu

	Vozidla	**Chodci**
Dopravní špička	1158	246
Běžný provoz	203	42

4.3.6 Simulace

Na vymodelovanou světelnou křižovatku se musel nachystat nástroj pro její spouštění (viz příloha *runner.py*). Vše byla napsáno v programovacím jazyce Python, s pomocí knihoven traci a sumolib, díky kterým se nadefinovala celková síť a byla umožněna kontrola simulace. Díky této kontrole bylo možno zadávat přesné trvání jednotlivých fází.

Spouštěcí funkce byla nastavena jako:

$$f(x), \tag{4.1}$$

kde x ∈ X a představuje časový údaj trvání fází. Pro potřeby optimalizace bylo x nastaveno jako x ∈ $<5, 90>$. Velikost tohoto bodu je:

$$[x_1, x_2, \dots, x_{15}]. \tag{4.2}$$

4.3.7 Výstupy simulace

Program SUMO podporuje velké množství výstupů, které shromažďují a zapisují do souborů ve formátu xml. Existuje však možnost převedení dat do csv souboru, to zde ale nebylo využito. Je třeba je ale explicitně definovat. Bylo nastaveno získávání dat o jednotlivých účastnících silničního provozu (tripinfo) a data celkové statistiky simulace (statistic).

Statistic

Celkové statistky simulace [26]:

- počet vozidel,
- počet chodců,
- průměrná délka trasy [m],
- průměrná cestovní rychlost [m/s],
- průměrná doba trvání cesty [s],
- celková doba jízdy všech vozidel [s],
- celkové zpoždění odjezdu všech vozidel [s] atd.

Zde je nejdůležitější parametr, pro který se hledá minimu, a to je celkové zpoždění odjezdu všech vozidel. V případě potřeby je však možnost hledat minimum pro jakýkoliv zmíněný parametr v tomto výstupu, kromě počtu účastníků silničního provozu, kteří jsou nastaveny pevně.

Tripinfo

Informace o každém vozidle a chodci, kteří byli použity v rámci simulace. Jsou to například [26]:

- doba vozidla strávená na cestě [s],
- čekací čas vozidla [s],
- časová ztráta vozidla [s] – čas jízdy pod ideální rychlostí,

- doba chodce strávená na cestě [s],
- časová ztráta chodce [s] atd.

4.4 Příprava optimalizačních metod

Pro optimalizaci, tedy hledání minima celkového zpoždění odjezdu všech vozidel, byly použity metody genetického algoritmu, Powellova metoda a Nelderova-Meadova metoda.

4.4.1 Popis použitého genetický algoritmus

Genetický algoritmus, dále jen GA, se ve výzkumech ohledně optimalizace dopravních sítí poměrně často vyskytuje [7]. Proto byla tato metoda zvolena, aby byl použit klasický algoritmus pro tento problém. Kód navrhnutého algoritmu je zahrnut v příloze (*genetic_algorithm.py*).

GA byl naprogramován v jazyce Python a jako inspirace posloužil [34]. Jeho algoritmus je následující. Nejprve se definují vstupní proměnné:

- počet generací (iterací),
- počet parametrů – na této konkrétní křižovatce to je 15,
- velikost populace,
- rozsah parametrů – zde pevně daný od 5 do 90,
- pravděpodobnosti křížení a mutace.

Celý proces byl již znázorněn na obrázku 5. Po spuštění GA se nejprve náhodně inicializovala počáteční populace v počtu 15ti parametrů (časových údajů jednotlivých fází) v rozsahu od 5ti do 90ti. Následně se na prvního rodiče v populaci spustila simulace, čímž se získalo první ohodnocení. Poté začala iterace, při které se pro každého rodiče v populaci spustila simulace a tím získala celá populace ohodnocení.

Použitá selekce, která následovala, byla typu elitního turnaje. Zde se vybral z ohodnocené populace rodič s nejlepším výsledkem (tedy nejmenší hodnotou). Zbytek populace se potom vybral náhodně.

Následovalo křížení nově vzniklé populace, které proběhlo na základě pravděpodobnosti. Při něm došlo k výměně chromozomů. Nakonec proběhla, opět na základě pravděpodobnosti, mutace. Pokud mutace proběhla, došlo k výměně chromozomu u rodiče.

Selekcí, křížením a mutací vznikly z rodičů děti, které se potom vzali do nové iterace a celý proces se opakoval, dokud nedošlo k naplnění zadaného počtu generací.

Celý průběh GA se zapisoval do textového souboru včetně běhu algoritmu a hledaných řešení (viz příloha *complet_final_result*).

4.4.2 Popis použití Powellovy a Nelderovy-Meadovy metody

Tyto metody, které se řadí ke starším optimalizačním metodám, se často pro optimalizaci dopravní sítě nepoužívají. Jsou však univerzální pro hledání minima a jejich implementace nebyla složitá, proto byly zvoleny. Byl pro ně rovněž použit programovací jazyk Python, a především jeho knihovna scipy, konkrétně její funkce minimize. V ní je jednou z možností zvolit si jako parametr konkrétní minimalizační metodu.

Pro spuštění těchto dvou metod bylo třeba do vstupních parametrů zadat již konkrétní časy jednotlivých fází a kolem nich se potom hledalo minimum. Jako tyto časy byly vybrány reálně naměřené hodnoty řešené konkrétní křižovatky (konkrétní hodnoty viz níže). Stejně jako u GA byl rozsah hledaných hodnot od 5ti do 90ti. Nalezené řešení se opět zapisovalo do textového souboru. K těmto metodám je kód opět v příloze (*optimize_algorithms.py*).

4.5 Spouštění simulace

Vyvinutý program byl koncipován tak, aby byla možnost spustit simulaci jak z příkazové řádky, tak ze simulačního prostředí programu SUMO. Simulace probíhají vždy dvojí, nejprve pro dopravní špičku a poté pro běžný provoz. Pro použití optimalizačních metod je výrazně doporučeno spouštět simulaci pouze v příkazovém řádku.

Pro možnosti porovnání nalezeného řešení s reálnou situací a taky s fázemi, které původně navrhl program SUMO, byla implementována i možnost spustit tyto dvě konkrétní situace. Dále je na výběr spustit simulaci s vlastními libovolnými časy jednotlivých fází. Na hlavní spouštění je program v příloze (*main.py*). Ukázka simulace v prostředí programu SUMO je v příloze *simulation.avi*.

5 ZHODNOCENÍ VÝSLEDKŮ

Zde budou postupně představeny jednotlivé výsledky a porovnání použitých metod. Proběhlo porovnání genetických algoritmů s různými nastaveními počátečních podmínek, dále srovnání nejlepších řešení pro optimalizační metody a ty se ještě porovnali s reálným scénářem a fázemi nabídnutými programem SUMO. Vše jak pro dopravní špičku, tak pro běžný provoz. Jak již bylo vícekrát řečeno hledala se minimální hodnota parametru celkové zpoždění odjezdu všech vozidel. Všechny výsledky, včetně grafů jsou v příloze v */complet_final_result.*

5.1 Porovnání genetických algoritmů

V této kapitole je srovnání použitého genetického algoritmu, který byl spuštěn s různými počátečními parametry (tabulka 6). Srovnání výsledků je potom v tabulce následující, tedy tabulka 7.

Tab. 6: Počáteční parametry genetického algoritmu

Název	Počet generací	Velikost populace	Počet parametrů	Pravd. křížení	Pravd. mutace	Rozsah hodnot
Alg_1	5	15	15	0,9	0,001	5-90
Alg_2	10	30				
Alg_3	10	50				
Alg_4	20	50				
Alg_5	5	100				

Tab. 7: Srovnání výsledků pro genetický algoritmus

	Dopravní špička		Běžný provoz	
Název	Doba běhu [s]	Celkové zpoždění [s]	Doba běhu [s]	Celkové zpoždění [s]
Alg_1	1 727	846 390	234	22 737
Alg_2	5 870	815 755	882	22 666
Alg_3	10 155	825 707	1 539	22 513
Alg_4	40 960	811 098	3 316	22 606
Alg_5	9 153	823 300	1 486	22 603

Jak se dalo předpokládat, je z výše uvedeného je patrné, že zvětšováním počtu generací i velikosti populace se celkové zpoždění zmenšuje to platí zejména u simulace při dopravní špičce. U běžného provozu se nejmenšího celkového zpoždění dosáhlo u Alg_3. Je to zřejmě z toho důvodu, že v dopravní síti nebylo dostatečně velké množství vozidel a chodců.

GA je založen na náhodě a pravděpodobnosti, což znamená, že při zvětšování počtu populací a generací se dochází k lepším výsledkům. Má to však negativní dopad na celkový čas běhu algoritmu.

5.2 Získání reálných dat

Experimentálně byly naměřeny hodnoty pro reálnou situaci. Uvedené časy nejsou korektně přesné, ale přibližné, blížící se realitě. Dopravní špička byla měřeny kolem třetí hodiny odpolední běžného pracovního dne, kdy je provoz vysoce frekventovaný a tvoří se dopravní zácpy ve všech směrech světelné křižovatky. Běžný provoz byl měřen okolo 10 hodiny dopolední pracovního dne. V tento čas se netvořily žádné kolony, ale provoz byl stále frekventovaný. Dále byly pro srovnání použity hodnoty navrhnuty programem SUMO, které byly vytvořeny při vygenerování dopravní sítě, viz kapitola 4.2.1. Vše zmíněné se nalezne v tabulce 8.

Tab. 8: Časování fází pro reálnou situaci

			Realita – dopravní špička	**Realita – běžný provoz**	**SUMO**
		Fáze	Trvání [s]		
Hlavní semafor		1	40	45	39
		2	10	25	9
		5	18	20	17
		8	15	47	15
		9	15	49	15
		12	25	22	24
		13	5	31	5
Vedlejší semafory	1T	1t	40	60	40
		3t	50	37	50
	2T	1t	40	24	41
		3t	50	36	50
	3T	1t	40	58	40
		3t	50	75	50
	4T	1t	40	78	39
		3t	50	47	49

5.3 Porovnání nalezených řešení pro dopravní špičku

V této sekci je porovnání jednotlivých optimalizačních metod pro dopravní špičku. Došlo k porovnání nejlepšího řešení nalezeného genetickým algoritmem, s Powellovou a Nelder-Meadovou metodou (tabulka 9). Navíc ještě porovnání s realitou a fázemi z programu SUMO.

Tab. 9: Porovnání optimalizované hodnoty jednotlivých metod pro dopravní špičku

Metoda	Celkové zpoždění [s]	Doba běhu [s]
GA	811 098	40 960
Powell	839 695	2 376
Nelder-Mead	845 647	1 004
Realita	860 022	-
SUMO	902 160	-

V tabulce 9 lze vidět, že nejlepšího výsledku, tedy nalezení minima celkového dopravního zpoždění, dosáhla metoda genetického algoritmu. Optimalizace však u něj běžela mnohonásobně delší dobu (přes 11 hodin) než u ostatních metod.

Srovnatelně dopadli metody Powella a Nelderova-Meadova, kdy Powell sice dosáhl lepšího výsledku, avšak jeho doba optimalizace byla zhruba dvakrát vyšší. Dalo se očekávat, že SUMO bude mít nejhorší výsledek, jelikož při generování dopravní sítě nezohledňuje optimální časy. Celková doba zpoždění u reálné situace nebyla příliš optimální, tudíž využití nástroje na optimalizaci se jeví jako ideální.

Ještě došlo k porovnání ostatních parametrů, které nebyly předmětem optimalizace, a to průměrná cestovní rychlost, průměrná doba trvání cesty a celková doba jízdy všech vozidel, viz tabulka 10.

Tab. 10: Porovnání neoptimalizované hodnoty jednotlivých metod pro dopravní špičku

Metoda	Cestovní rychlost [m/s]	Doba trvání cesty [s]	Doba jízdy všech vozidel [s]
GA	5,57	120	139 619
Powell	4,45	741	202 920
Nelder-Mead	6,13	121	140 528
Realita	6,14	124	144 445
SUMO	3,12	261	302 336

Z tabulky je zřejmé, že nalezené časování fází nejlepším algoritmem, tedy GA, bylo i pro neoptimalizované hodnoty nejlepší variantou. Výjimku tvoří jen parametr cestovní rychlost, který vyšel lépe pro fáze nastaveny programem SUMO a pro fáze Powellovy metody. Cestovní rychlost je však z hlediska dopravní špičky méně podstatná

než doba trvání cesty a doba jízdy všech vozidel. Z výše zmíněného vyplývá, že nastavení fází podle GA, se jeví jako nejlepší varianta.

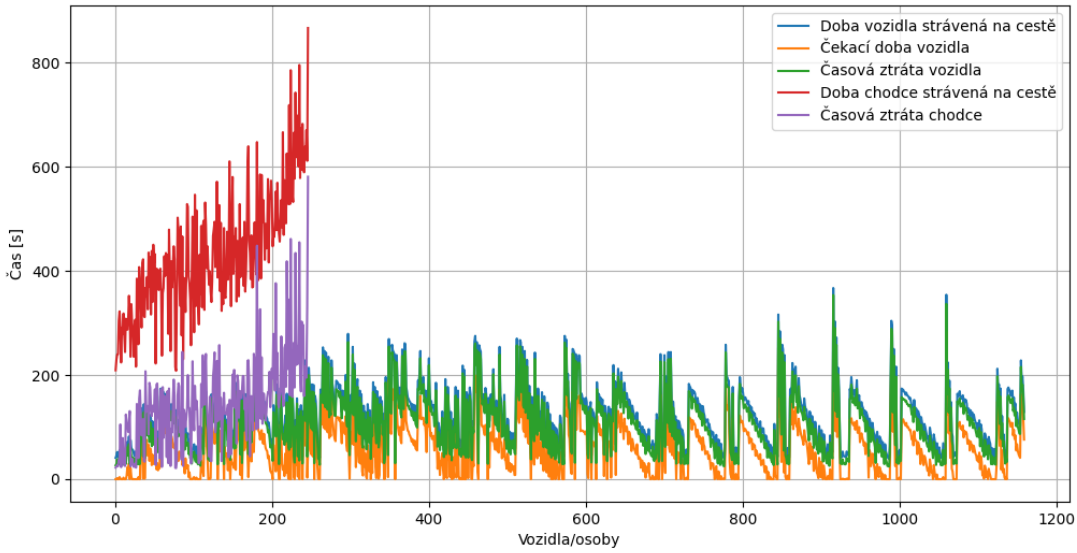
Výsledné časování fází je v tabulce 11. Zajímavostí je, že se všechny časové údaje pro jednotlivé metody liší. To může být způsobeno velkou variabilitou možností. Ještě stojí za povšimnutí, že genetický algoritmus jednotlivé časové úseky rozdělil zhruba na stejný časový údaj. Oproti tomu zbylé dvě metody měly převážně pro vedlejší semafory dlouhé úseky.

Tab. 11: Časování fází pro dopravní špičku

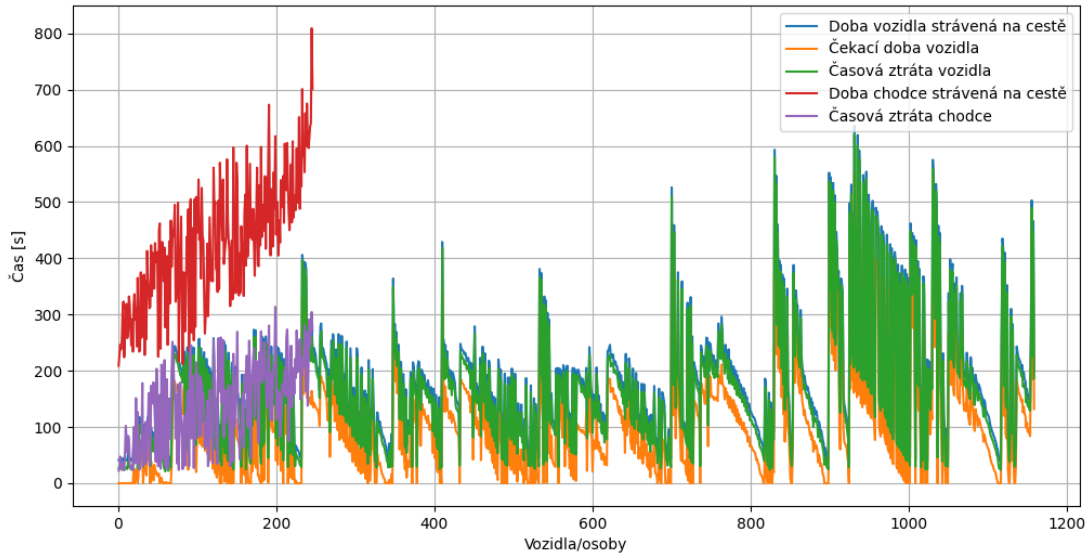
			GA	**Powell**	**Nelder-Mead**
		Fáze	Trvání [s]		
Hlavní semafor		1	12	45	39
		2	26	25	9
		5	14	20	17
		8	23	47	15
		9	20	49	15
		12	20	22	24
		13	25	31	5
Vedlejší semafory	1T	1t	15	60	40
		3t	11	37	50
	2T	1t	37	24	41
		3t	1	36	50
	3T	1t	23	58	40
		3t	1	75	50
	4T	1t	19	78	39
		3t	4	47	49

5.4 Informace o dopravní cestě pro dopravní špičku

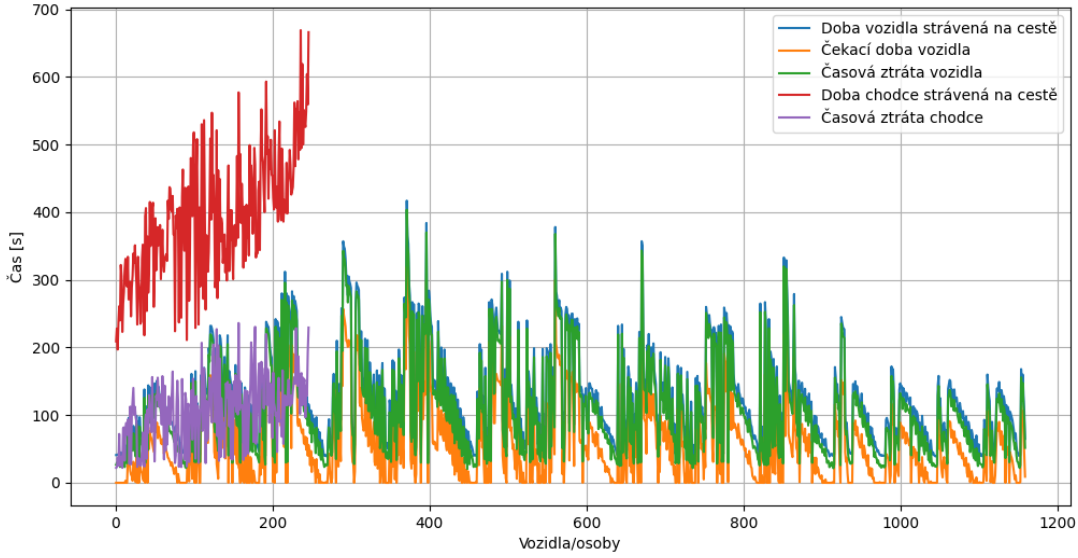
Ve výstupu po simulaci světelné křižovatky jsou i informace o každém účastníkovi silničního provozu během simulace. Byly vybrány parametry z kapitoly 4.3.7 nadpis tripinfo, které sloužily pro srovnání se všemi použitými metodami. V Pythonu za pomocí knihovny Matplotlib byly vytvořeny grafy (obrázky 20 až 24.), které znázorňují zmiňované parametry v čase pro účastníky silničního provozu. Grafy byly vytvořeny v programu Python pomocí knihovny Matplotlib (příloha *statistisc.py*).



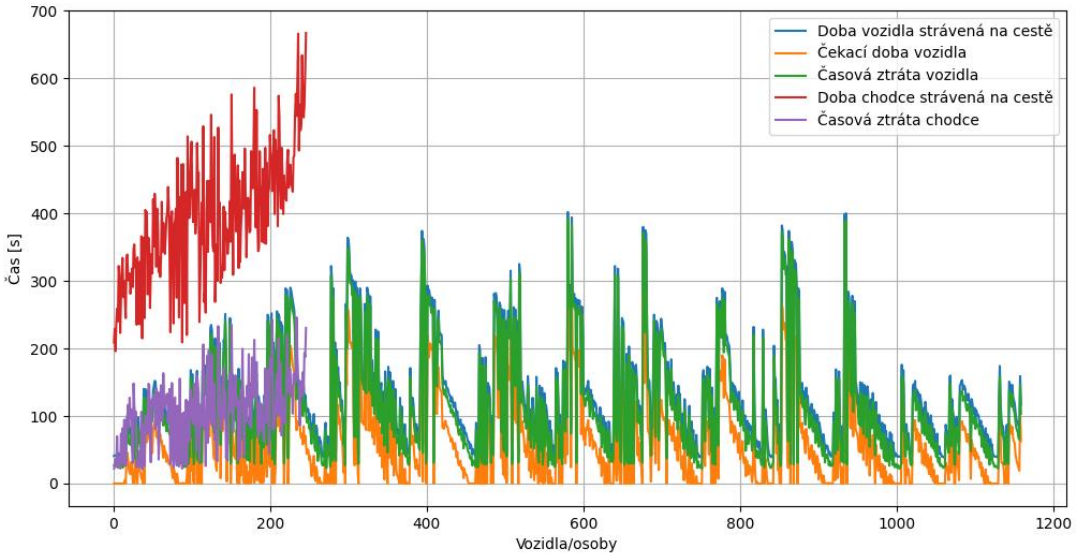
Obr. 20: Informace o dopravní cestě pro dopravní špičku – GA



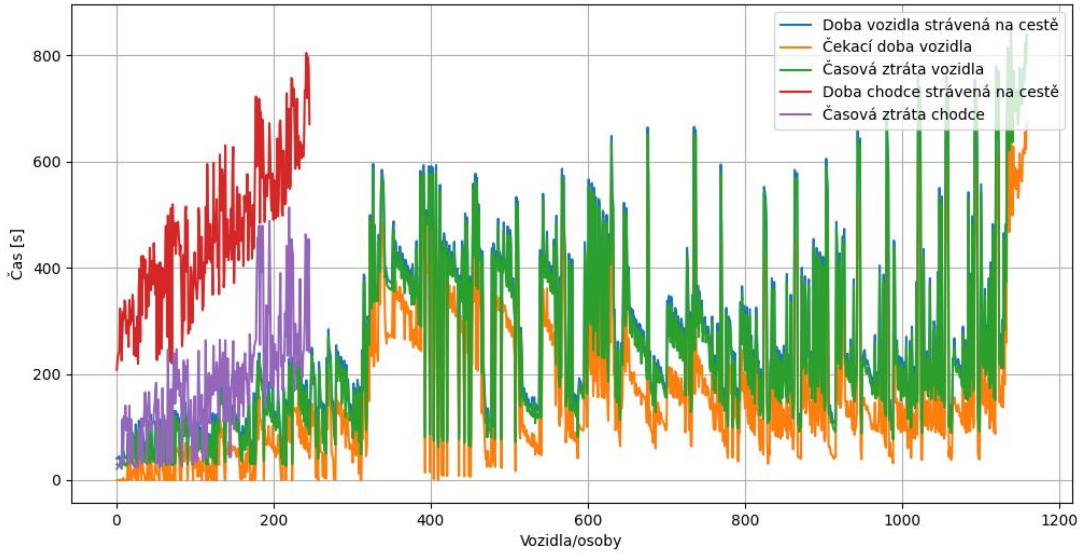
Obr. 21: Informace o dopravní cestě pro dopravní špičku – Powell



Obr. 22: Informace o dopravní cestě pro dopravní špičku – Nelder-Mead



Obr. 23: Informace o dopravní cestě pro dopravní špičku – reálná situace



Obr. 24: Informace o dopravní cestě pro dopravní špičku – SUMO

Pokud se zaměříme na jednotlivé parametry a jaká metoda v nich byla nejúspěšnější, tak potom:

- doba vozidla strávená na cestě – GA,
- čekací doba vozidla – GA,
- časová ztráta vozidla – GA,
- doba chodců strávená na cestě – Nelder-Mead a reálná situace,
- časová ztráta chodce – Nelder-Mead a reálná situace.

Když se vezmou do úvahy jenom použité optimalizační metody, tak pro vozidla se jako nejlepší jeví nastavení fází podle GA. Pro chodce to jsou fáze podle Nelder – Meada.

Ukázalo se, že metoda podle Nelder-Meada hledala optimální řešení jen v těsné blízkosti bodů daných reálnou situací.

5.5 Porovnání nalezených řešení pro běžný provoz

Podobně jako pro dopravní špičku, tak i pro běžný provoz bylo provedeno srovnání optimalizačních metod (viz tabulka 12) stejným způsobem jako v kapitole 5.3.

Tab. 12: Porovnání jednotlivých metod pro běžný provoz

Metoda	Celkové zpoždění [s]	Doba běhu [s]
GA	22 513	1 539
Powell	22 913	1 022
Nelder-Mead	22 819	347
Realita	22 870	-
SUMO	23 020	-

Obdobně jako v případě dopravní špičky, i při běžném provozu vyšla optimalizace pomocí genetického algoritmu nejlépe. Opět s negativem, a to nejdelším během algoritmu (asi 25 minut).

Nejvíce optimální se pro tento problém jeví optimalizace pomocí metody Nelder-Meada. Ta sice nenalezla nejmenší celkové dopravní zpoždění, avšak její doba běhu byla více něž čtyřikrát rychlejší oproti GA.

U běžného provozu se oproti dopravní špičce neprojevil tak velký rozdíl mezi reálnou situací.

Dále došlo k porovnání ostatních parametrů, které nebyly optimalizovány, avšak určitě stojí za zmínku. Jsou to průměrná cestovní rychlost, průměrná doba trvání cesty a celková doba jízdy všech vozidel, viz tabulka 10.

Tab. 13: Porovnání neoptimalizované hodnoty jednotlivých metod pro běžný provoz

Metoda	Cestovní rychlost [m/s]	Doba trvání cesty [s]	Doba jízdy všech vozidel [s]
GA	5,92	131	26 594
Powell	4,53	232	47 125
Nelder-Mead	6,47	100	20 302
Realita	6,47	100	20 302
SUMO	7,67	81	16 443

Zde je vidět, že nejlepší nastavení časování fází bylo základní vytvořené programem SUMO při generování dopravní sítě. GA, nejlepší algoritmus pro minimalizaci celkového dopravní zpoždění, se na těchto parametrech nejevil nejlépe.

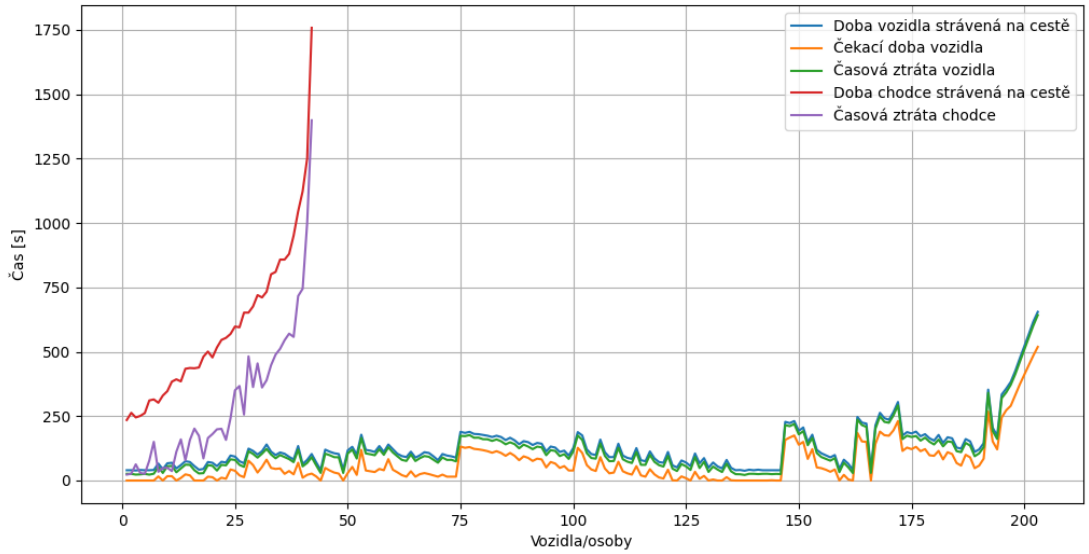
V tabulce 12 se nachází výsledné časování fází pro všechny metody optimalizace. Stojí za povšimnutí, že genetický algoritmus vyhodnotil trvání jedné fáze na 0. Jedná se o fázi přechodu pro chodce. Je to zřejmě způsobeno tím, že při náhodném generování chodců a jejich tras se žádná z nich neprotnula s tímto semaforem.

Tab. 14: Časování fází pro běžný provoz

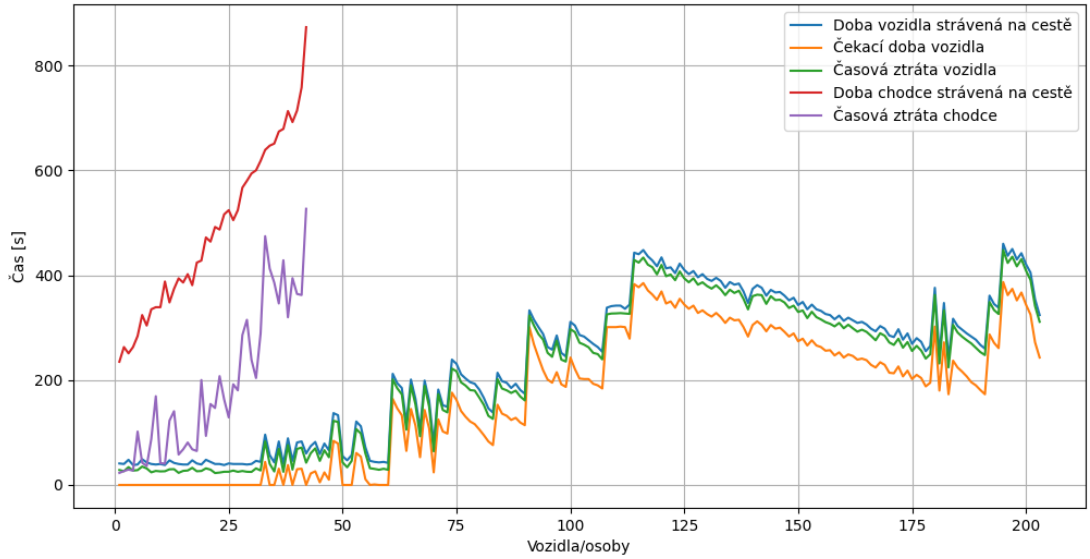
			GA	**Powell**	**Nelder-Mead**
		Fáze	Trvání [s]		
Hlavní semafor		1	33	57	30
		2	40	57	10
		5	23	69	18
		8	26	82	13
		9	3	57	12
		12	45	89	13
		13	19	89	12
Vedlejší semafory	1T	1t	4	69	35
		3t	47	37	30
	2T	1t	26	59	35
		3t	23	37	30
	3T	1t	0	31	35
		3t	38	67	30
	4T	1t	30	37	35
		3t	16	46	30

5.6 Informace o dopravní cestě pro běžný provoz

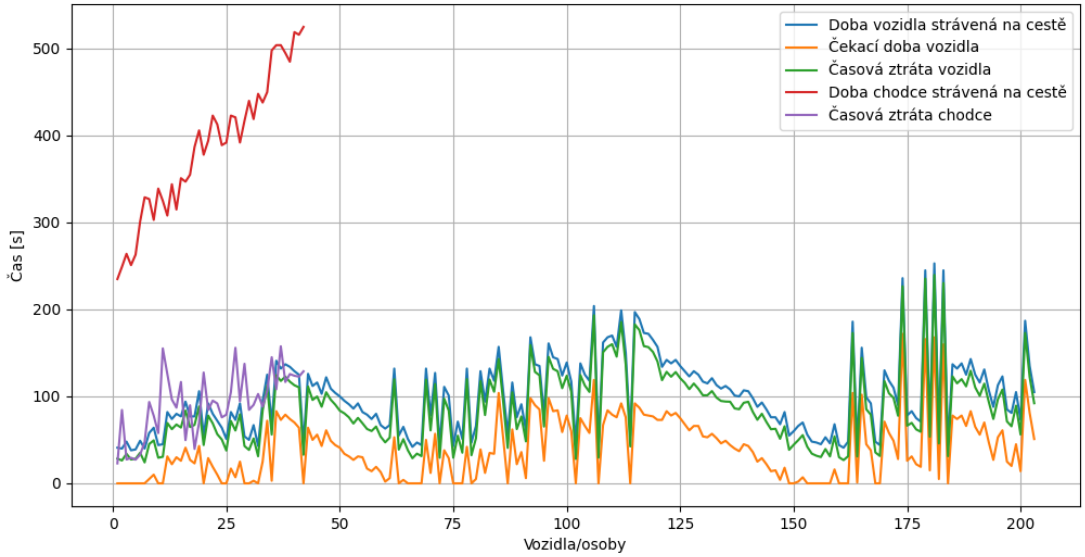
I pro běžný provoz byly ve výstupu po simulaci světelné křižovatky informace o každém vozidle a chodci, kteří se objevili v simulované dopravní síti. Byly vybrány stejné parametry, jako v kapitole 5.4. a vytvořeny grafy (obrázky 25 až 29.),



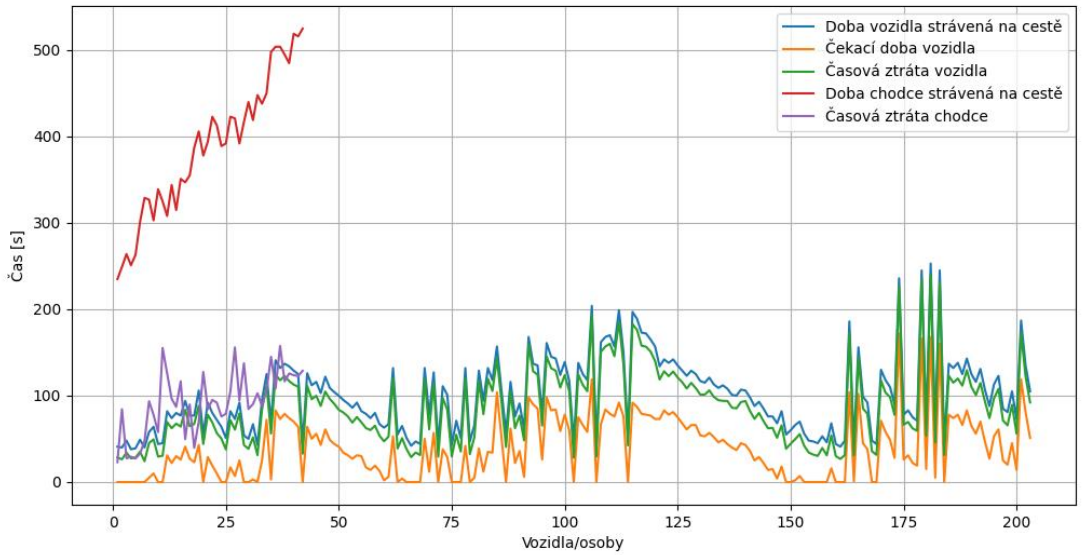
Obr. 25: Informace o dopravní cestě pro běžný provoz – GA



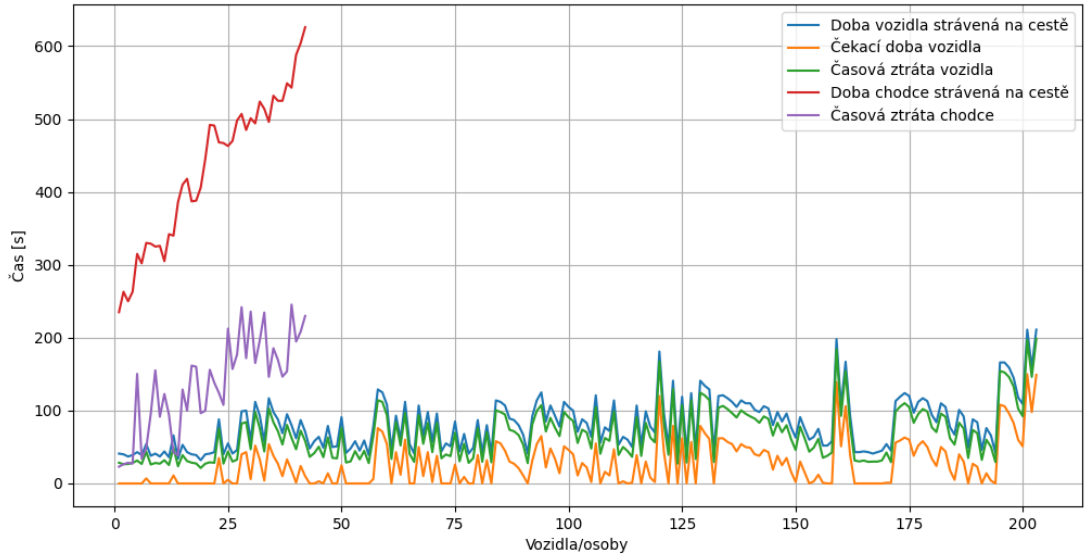
Obr. 26: Informace o dopravní cestě pro běžný provoz – Powell



Obr. 27: Informace o dopravní cestě pro běžný provoz – Nelder-Mead



Obr. 28: Informace o dopravní cestě pro běžný provoz – reálná situace



Obr. 29: Informace o dopravní cestě pro běžný provoz – SUMO

Pokud se opět zaměříme na jednotlivé parametry a jaká metoda v nich byla nejúspěšnější, tak potom:

- doba vozidla strávená na cestě – SUMO,
- čekací doba vozidla – SUMO,
- časová ztráta vozidla – SUMO,
- doba chodců strávená na cestě – Nelder-Mead a reálná situace,
- časová ztráta chodce – Nelder-Mead a reálná situace.

Pro běžný provoz se jako ideální nastavení časování fází pro vozidla ukazuje nastavení dané programem SUMO. Jako optimalizační algoritmus by to potom byla metoda Nelderova-Meadova. Ta samá metoda je i nejvíce vhodná pro chodce.

5.7 Prohození optimálních řešení

V obou případech, tedy pro dopravní špičku i běžný provoz, byla nejmenší hodnota celkového zpoždění vozidel, nalezena genetickým algoritmem. Trvání fází navrhnutých GA by tedy z hlediska tohoto parametru nejideálnější. Fáze jsou shrnuty v tabulce 15.

Tab. 15: Doba trvání fází pro nejlepší nalezené řešení

	Hlavní semafor							**Vedlejší semafory**							
								1T		2T		3T		4T	
Fáze	1	2	5	8	9	12	13	1t	3t	1t	3t	1t	3t	1t	3t
Dopravní špička	12	26	14	23	20	20	25	15	11	37	1	23	1	19	4
Běžný provoz	33	40	23	26	3	45	19	4	47	26	23	0	38	30	16

Jednotlivé fáze byly následně prohozeny tak, že fáze určená pro dopravní špičku byla implementována na simulaci při běžném provozu a naopak. Následně byla spuštěna simulace a došlo se k následujícím výsledkům (tabulka 16).

Tab. 16: Srovnání prohozených nejlepších fází

	Celkové zpoždění [s]	**Cestovní rychlost [m/s]**	**Doba trvání cesty [s]**	**Doba jízdy všech vozidel [s]**
Dopravní špička	969 464	4,80	246	285 598
Běžný provoz	22 729	6,51	101	20 642

Když se parametry z tabulky srovnají s ostatními dojde ke zjištění, že fáze, které jsou určené primárně pro běžný provoz jsou pro dopravní špičku nepoužitelné. Výsledné hodnoty této simulace byly mnohokrát horší než původní.

Naopak fáze, které jsou primárně určeny pro dopravní špičku, se mohou aplikovat i pro běžný provoz. Jejich řešení sice není lepší než původní, ale je s tím původním srovnatelné.

Z výše uvedeného vyplývá, že v případě řízení s pevným časem, které má pevnou délku cyklu po celou dobu, jsou získané fáze nastavené pro dopravní špičku vhodné. Naopak fáze nastavené na běžný provoz jsou pro tento stav řízení nepoužitelné.

6 ZÁVĚR

Práce se zabývala problémem řízení světelné křižovatky. Cílem bylo tento problém popsat a provést průzkum na aktuálně používané metody a následně vytvořit softwarovou implementaci.

V první části byly popsány problémy řízení světelné křižovatky tzn. představeny vstupní a výstupní parametry této křižovatky, jako i její druhy a dělení. Poté se přešlo k popisu optimalizačních metod pro tento problém, konkrétně genetický algoritmus, Petriho sítě, Q-učení a numerické metody.

Praktická část spočívala ve vymodelování dopravní sítě dle konkrétní křižovatky a na ni poté, pomocí simulace, byly aplikovány optimalizační metody. Pro tyto účely byla vybrána světelná křižovatka ve městě Olomouc. Na optimalizaci byly zvoleny následující: genetický algoritmus, Powellova metoda a Nelderova-Meadova metoda. Simulaci a světelnou křižovatku pomohl zprostředkovat program SUMO. Optimalizovalo se nastavení trvání fází semaforů na konkrétní křižovatce, přičemž se hledalo minimum pro parametr „celkové dopravní zpoždění všech vozidel“. Byl vytvořen nástroj na tvorbu sítě světelné křižovatky a na spouštění simulace pomocí optimalizačních metod, ale také na spouštění s vlastní volbou trvání jednotlivých fází světelné křižovatky.

Optimalizace proběhla na dva druhy silničního provozu, a to byly dopravní špička a běžný provoz. Hustota provozu těchto variant byla generována automaticky na základě pravděpodobnosti. Simulace byla spouštěna i s nastavením časování fází pro reálnou situaci, kdy tato data byla získána měřením časování na konkrétní křižovatce. A poté ještě s nastavením fází, které byly vytvořeny automaticky programem SUMO při generování sítě.

Následně došlo k porovnání použitých optimalizačních metod a jednotlivých simulací. Pro dopravní špičku i běžný provoz vyšlo jednoznačně nejlepší trvání fází nalezené genetickým algoritmem. Jeho nevýhodou oproti dalším použitým metodám však byla jeho mnohonásobně vyšší doba běhu optimalizace. Metodami Powell a Nelder-Mead však také bylo nalezeno řešení ideálnější než u reálné situace. Výjimku tvoří akorát metoda Powell při optimalizaci běžného provozu.

Dále byly srovnávány další parametry simulace světelné křižovatky, které nebyly předmětem optimalizace a týkaly se všech účastníků silničního provozu na vymodelované křižovatce. Jsou to průměrná cestovní rychlost, průměrná doba trvání cesty a celková doba jízdy všech vozidel. Zde u dopravní špičky vyšel jako nejlepší opět genetický algoritmus. U běžného provozu však měla nejideálnější výsledky simulace spuštěná s nastavením fází podle programu SUMO. Jako optimalizační metoda u běžného provozu dopadla nejlépe metoda Nelder-Mead.

Ještě byly porovnány hodnoty získané pro jednotlivé účastníky silničního provozu. Tyto hodnoty jsou: doba vozidla strávená na cestě, čekací čas vozidla, časová ztráta vozidla, doba chodce strávená na cestě a časová ztráta chodce. Srovnání dopravní špičky i běžného provozu dopadlo stejně a to následovně. Pro vozidla bylo nejlepší

nastavení fází podle genetického algoritmu, s ohledem na chodce to však byla metoda Nelder-Meada.

V posledním experimentu došlo k prohození nalezených nejlepších řešení časování fází a to následovně. Fáze určená pro dopravní špičku byla implementována na simulaci při běžném provozu a naopak. Bylo zjištěno, že fáze určené pro dopravní špičku jsou použitelné i pro běžný provoz. V opačném případě to tak neplatí.

Ze všech srovnání vyplývá, že nejlepší použitá metoda byl genetický algoritmus. S nastavením ještě vyšších počátečních parametrů jako jsou velikost populace a počet generací by bylo téměř jistě dosaženo ještě lepších výsledků. Časová náročnost by byla ale velká. Jistě by stálo za zvážení jej aplikovat i na situaci, kde by byla hustota provozu podle skutečných dat, a nejen vytvořena na základě pravděpodobnosti. Navržený nástroj by po získání skutečných dat a drobné úpravě tuto situaci dokázal nasimulovat.

7 SEZNAM POUŽITÉ LITERATURY

1] CHEN, Lei a Cristofer ENGLUND. Cooperative Intersection Management: A Survey. *IEEE Transactions on Intelligent Transportation Systems*. 2016, **17**(2), 570-586. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2015.2471812

[2] QADRI, Syed Shah Sultan Mohiuddin, Mahmut Ali GÖKÇE a Erdinç ÖNER. State-of-art review of traffic signal control methods: challenges and opportunities. *European Transport Research Review*. 2020, **12**(1). ISSN 1867-0717. Dostupné z: doi:10.1186/s12544-020-00439-1

[3] *TECHNICKÉ PODMÍNKY – TP 81 Navrhování světelných signalizačních zařízení pro řízení provozu na pozemních komunikacích*. ČDS ČR, 3. vydání, 2015. Dostupné z: http://www.pjpk.cz/data/USR_001_2_8_TP/TP_81.pdf

[4] LIST, G.F. a M. CETIN. Modeling Traffic Signal Control Using Petri Nets. *IEEE Transactions on Intelligent Transportation Systems*. 2004, **5**(3), 177-187. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2004.833763

[5] SMĚLÝ, Martin. *Dopravní inženýrství: Řízené úrovňové křižovatky I*. Brno, 2007. Studijní podpora. VUT v Brně

[6] PŘIBYL, Pavel a Miroslav SVÍTEK. *Inteligentní dopravní systémy*. Praha: BEN – technická literatura, 2001. ISBN 80-730-0029-6.

[7] EOM, Myungeun a Byung-In KIM. The traffic signal control problem for intersections: a review. *European Transport Research Review*. 2020, **12**(1). ISSN 1867-0717. Dostupné z: doi:10.1186/s12544-020-00440-8

[8] LEE, Seunghyeon, S.C. WONG a Pravin VARAIYA. Group-based hierarchical adaptive traffic-signal control part I: Formulation. *Transportation Research Part B: Methodological*. 2017, **105**, 1-18. ISSN 01912615. Dostupné z: doi:10.1016/j.trb.2017.08.008

[9] SMITH, M.J. Traffic control and route-choice; a simple example. *Transportation Research Part B: Methodological*. 1979, **13**(4), 289-294. ISSN 01912615. Dostupné z: doi:10.1016/0191-2615(79)90021-3

[10] ALONSO-AYUSO, Antonio, Laureano F. ESCUDERO a F. Javier MARTIN-CAMPO. Collision Avoidance in Air Traffic Management: A Mixed-Integer Linear Optimization Approach. *IEEE Transactions on Intelligent Transportation Systems*. 2011, **12**(1), 47-57. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2010.2061971

[11] TOMAS-GABARRON, Juan-Bautista, Esteban EGEA-LOPEZ a Joan GARCIA-HARO. Vehicular Trajectory Optimization for Cooperative Collision Avoidance at High Speeds. *IEEE Transactions on Intelligent Transportation Systems*. 2013, **14**(4), 1930-1941. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2013.2270009

[12] KOCHENDERFER, Mykel J. a Tim A. WHEELER. *Algorithms for optimization*. Cambridge, Massachusetts: The MIT Press, [2019]. ISBN 978-026-2039-420.

[13] ALBADR, Musatafa Abbas, Sabrina TIUN, Masri AYOB a Fahad AL-DHIEF. Genetic Algorithm Based on Natural Selection Theory for Optimization Problems. *Symmetry*. 2020, **12**(11). ISSN 2073-8994. Dostupné z: doi:10.3390/sym12111758

[14] Crossover Operators in Genetic Algorithm. *Medium* [online]. [cit. 2022-05-20]. Dostupné z: https://medium.com/geekculture/crossover-operators-in-ga-cffa77cdd0c8

[15] ABDULHAI, Baher, Rob PRINGLE a Grigoris J. KARAKOULAS. Reinforcement Learning for True Adaptive Traffic Signal Control. *Journal of Transportation Engineering*. 2003, **129**(3), 278-285. ISSN 0733-947X. Dostupné z: doi:10.1061/(ASCE)0733-947X(2003)129:3(278)

[16] What Is Q-Learning: The Best Guide To Understand Q-Learning. *Simplilearn* [online]. [cit. 2022-05-20]. Dostupné z: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning#what_is_qlearning

[17] WATKINS, Christopher J. C. H. a Peter DAYAN. Q-learning. *Machine Learning*. 1992, **8**(3-4), 279-292. ISSN 0885-6125. Dostupné z: doi:10.1007/BF00992698

[18] DIFEBBRARO, A., D. GIGLIO a N. SACCO. Urban Traffic Control Structure Based on Hybrid Petri Nets. *IEEE Transactions on Intelligent Transportation Systems*. 2004, **5**(4), 224-237. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2004.838180

[19] ŠŤASTNÝ, Jiří, Michal JEŽEK a Kamil STANĚK. *Simulace dynamických systémů*. Brno, 2020. Studijní podpora. VUT v Brně.

[20] ANDRZEJEWSKI, Grzegorz, Wojciech ZAJĄC, Kazimierz KRZYWICKI, Artur KARASIŃSKI, Tomasz KRÓLIKOWSKI a Błażej BAŁASZ. Implementation of an example of Hierarchical Petri Net (HPN) in LAD language in TIA Portal. *Procedia Computer Science*. 2021, **192**, 3657-3666. ISSN 18770509. Dostupné z: doi:10.1016/j.procs.2021.09.139

[21] ČERMÁK, Libor a Rudolf HLAVIČKA. *Numerické metody*. Vydání třetí. Brno: Akademické nakladatelství CERM, 2016. ISBN 978-80-214-5437-8.

[22] ANTONIOU, Andreas a WU-SHENG LU. *Practical Optimization: Algorithms and Engineering Applications* [online]. Second Edition. New York, 2021 [cit. 2022-05-20]. ISBN 978-1-0716-0843-2. Dostupné z: https://doi.org/10.1007/978-1-0716-0843-2

[23] POWELL, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*. 1964, **7**(2), 155-162. ISSN 0010-4620. Dostupné z: doi:10.1093/comjnl/7.2.155

[24] PETERSEN, S.J. *Methods of Optimization for Numerical Algorithms* [online]. Groningen, 2017 [cit. 2022-05-20]. Dostupné z: https://fse.studenttheses.ub.rug.nl/15741/2/BSC_Math_2017_Petersen_SJ.pdf. Bachelor's Project Mathematics. University of Groningen.

[25] *Mapy.cz* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://mapy.cz/

[26] LOPEZ, Pablo Alvarez, Evamarie WIESSNER, Michael BEHRISCH, et al. Microscopic Traffic Simulation using SUMO. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, 2018, 2575-2582. ISBN 978-1-7281-0321-1. Dostupné z: doi:10.1109/ITSC.2018.8569938

[27] *Python Software Foundation* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://www.python.org/

[28] *NumPy* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://numpy.org/

[29] *Matplotlib* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://matplotlib.org/

[30] Beautiful Soup Documentation. *Crummy* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/#

[31] The ElementTree XML API. *Python Software Foundation* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://docs.python.org/3/library/xml.etree.elementtree.html

[32] *SciPy* [online]. 2022 [cit. 2022-05-20]. Dostupné z: https://scipy.org/

[33] Světelná signalizace na křižovatkách. *Zelená vlna* [online]. 2022, 2007 [cit. 2022-05-20]. Dostupné z: https://zelenavlna.rozhlas.cz/svetelna-signalizace-na-krizovatkach-7920321

[34] Simple Genetic Algorithm From Scratch in Python. *Machine Learning Mastery* [online]. 2022, 2021 [cit. 2022-05-20]. Dostupné z: https://machinelearningmastery.com/simple-genetic-algorithm-from-scratch-in-python/

8 SEZNAM OBRÁZKŮ A TABULEK

8.1 Seznam obrázků

8.2 Seznam tabulek

9 PŘÍLOHY

Přiložený zip archiv obsahuje:

1) prepare_cross.py – vytvoření simulační sítě,
2) genetic_algorithm.py – nástroj genetického algoritmu,
3) optimize_algorithms.py – nástroj na spouštění Powellovy Nelder-Meadovy metody,
4) runner.py – nástroj na spouštění simulace,
5) main.py - hlavní program, ve kterém probíhá spouštění simulace,
6) statistics.py – vytvoření grafů z výsledných simulací,
7) cfg – adresář se konfiguračními soubory potřebnými pro spuštění simulace,
8) data_to_optimalized – uložená vymodelovaná síť, na kterou se spouštěla simulace v této diplomové práci,
9) complet_final_result – kompletní soubory s výsledky simulace této diplomové práce
10) simulation.avi – ukázka simulace v grafickém rozhraní