VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

KONVERGOVANÉ SÍTĚ A TOMOGRAFIE SÍŤOVÉHO PROVOZU S VYUŽITÍM EVOLUČNÍCH ALGORITMŮ
CONVERGED NETWORKS AND TRAFFIC TOMOGRAPHY BY USING EVOLUTIONARY ALGORITHMS

DIZERTAČNÍ PRÁCE
DOCTORAL THESIS

AUTOR PRÁCE
AUTHOR
Ing. Václav Oujezský

ŠKOLITEL
SUPERVISOR
doc. Ing. Vladislav Škorpil, CSc.

BRNO 2017

ABSTRAKT

Tomografie síťového provozu představuje dnes již nedílnou součást v oblasti konvergovaných sítí a systémů k detekci jejich behaviorálních vlastností. Dizertační práce se zabývá výzkumem její implementace s využitím evolučních algoritmů. Výzkum byl zejména soustředěn na inovaci a řešení behaviorální detekce toků dat v sítích a jejich anomálií s využitím síťové tomografie a evolučních algoritmů. V rámci řešení dizertační práce byl navržen nový algoritmus, vycházející ze základů statistické metody analýzy přežití v kombinaci s algoritmem genetickým. Navržený algoritmus byl testován ve vlastním vytvořeném modelu síťové sondy za pomocí programovacího jazyka Python a laboratorních síťových zařízení Cisco. Provedené testy prokázaly základní funkčnost navrženého řešení.

KLÍČOVÁ SLOVA

Behaviorální analýza sítě, evoluční algoritmus, Python, síťová sonda, tomografie síťového provozu.

SUMMARY

Nowadays, the traffic tomography represents an integral component in converged networks and systems for detecting their behavioral characteristics. The dissertation deals with research of its implementation with the use of evolutionary algorithms. The research was mainly focused on innovation and solving behavioral detection data flows in networks and network anomalies using tomography and evolutionary algorithms. Within the dissertation has been proposed a new algorithm, emerging from the basics of the statistical method survival analysis, combined with a genetics' algorithm. The proposed algorithm was tested in a model of a self-created network probe using the Python programming language and Cisco laboratory network devices. Performed tests have shown the basic functionality of the proposed solution.

KEYWORDS

Network Behavior Analysis, Evolutionary Algorithms, Network Probe, Python, Network Tomography.

OUJEZSKÝ, Václav *Konvergované sítě a tomografie síťového provozu s využitím evolučních algoritmů*: dizertační práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 111 s. Vedoucí práce byl doc. Ing. Vladislav Škorpil, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou doktorskou práci na téma „Konvergované sítě a tomografie síťového provozu s využitím evolučních algoritmů“ jsem vypracoval samostatně pod vedením vedoucího doktorské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené doktorské práce dále prohlašuji, že v souvislosti s vytvořením této doktorské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 11. června 2017 Václav Oujezský

PODĚKOVÁNÍ

Děkuji svému školiteli a vedoucímu mé dizertační práce panu doc. Ing. Vladislavu Škorpilovi, CSc. za jeho odborné vedení, konzultace, trpělivost, podnětné návrhy a profesionální přístup. Děkuji také Ústavu telekomunikací, který mi umožnil realizovat výzkum.
Neocenitelné poděkování patří celé mé rodině za její nekonečnou podporu, které jí tímto děkuji.

Brno 11. června 2017 Václav Oujezský

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této doktorské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno 11. června 2017 Václav Oujezský

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

Úvod **11**

1 Přehled současného stavu **13**
1.1 Konvergované sítě 13
1.2 Tomografie síťového provozu 15
1.2.1 Algoritmy grafu 16
1.3 Základní techniky detekcí anomálií v síťovém provozu 17
1.3.1 Techniky verifikace protokolů 17
1.3.2 Techniky statistického modelování a analýzy 18
1.3.3 Techniky výpočetní inteligence 24
1.3.4 Síťová tomografie v roli detekcí anomálií 25
1.3.5 Výčet dalších metod a technik 26
1.4 Získávání informací o síti pro účely analýzy 27

2 Teoretický úvod evolučních algoritmů **35**
2.1 Genetický algoritmus 38
2.1.1 Selekce 40
2.1.2 Křížení 42
2.1.3 Mutace 43
2.2 Typy genetických algoritmů 44
2.2.1 Paralelní GA a paralelní implementace 44

3 Stěžejní publikace k dané problematice **46**

4 Specifikace výzkumu **47**

5 Testování programů pro vývoj a implementaci algoritmu **51**
5.1 Simulační prostředí a pracovní nástroje 51
5.1.1 Srovnávací test programu OMNeT++ a Python 52

6 Testování hypotézy analýzy přežití **54**
6.1 Výsledky inicializačního testu 55
6.2 Výsledky testu podobnosti a modelu provozu 57

7 Testování evolučních algoritmů **60**
7.1 Ilustrace 1 – Maximalizace dané funkce 60
7.2 Ilustrace 2 – Výkonnost algoritmu 62

7.3 Ilustrace 3 – Problém batohu 63
7.3.1 Závěrečná zjištění a shrnutí ilustrací GA 65

8 Návrh algoritmu a modelu 66
8.1 Modul kolektor 67
8.1.1 Uživatelské rozhraní 67
8.2 Modul supervizor 69
8.2.1 Výčet dat z databáze a návrh implementace analýzy přežití . . 69
8.2.2 Shluková analýza a genetický algoritmus 72
8.2.3 Implementace algoritmu K-průměrů 73
8.2.4 Návrh a implementace genetického algoritmu 74
8.2.5 Výsledky implementace genetického algoritmu 79
8.3 Implementace genetického algoritmu v FPGA 83

9 Závěr a diskuze dosažených výsledků 85

Literatura 88

Seznam použitých zkratek 98

Seznam symbolů 102

Seznam příloh 103

A NetFlow verze 5 – vysvětlivky 104

B Přehled evolučních algoritmů 105

C Schéma genetického algoritmu FPGA 106

D Obsah přiloženého CD 107

Publikační činnost 108

Vedené závěrečné práce 110

Životopis 111

SEZNAM OBRÁZKŮ

1.1 Reprezentace síťové tomografie pomocí grafu uzlů 15
1.2 PCA analýza variance latence provozu ze síťových sond 21
1.3 Kontinuální ping sondou RIPE 27
1.4 Grafické výstupy ze síťových sond 28
1.5 Základní sekvenční schéma IDS 29
1.6 Základní schéma zapojení technologií pro monitoring 31
1.7 Graf PRTG sondy NetFlow verze 9 32
1.8 Formát NetFlow export datagramu verze 5 [81] 33
2.1 Základní diagram genetického algoritmu 39
2.2 Ruletová selekce 41
2.3 Porovnání rozdělení hustoty pravděpodobností 44
2.4 Paralelní zpracování GA 45
4.1 Princip pozorování událostí výskytu 48
4.2 Jednoduchá centralizovaná síť botnet 50
5.1 Základní schéma simulace v Python a OMNeT++ 53
6.1 Zapojení laboratorních prvků 54
6.2 Zobrazení zachycené komunikace v SQL databázi 56
6.3 Test Kaplan-Meier 56
6.4 Test Kaplan-Meier Botnet 58
6.5 Zobrazení křivek přežití pro jednotlivé protokoly 59
7.1 Ilustrace GA 1 – maximalizace 60
7.2 Ilustrace GA 1 – vliv operátorů 61
7.3 Pareto fronta – NSGAII. 62
7.4 Průběh GA – vliv velikosti populace 64
7.5 Průběh GA - vliv parametrů 64
8.1 Základní sekvenční schéma GDP 66
8.2 Grafické zobrazení aplikace 67
8.3 Rozložení databáze NetFlow 70
8.4 Analýza přežití pro všechna SD testovaného provozu 71
8.5 Zobrazení vstupních testovacích sad pro shlukovou analýzu 72
8.6 Třída Davies-Bouldin index 72
8.7 Třídní diagram K-Means.py 73
8.8 Třídní diagram GeneticsCore.py 76
8.9 Princip křížení chromozomů 77
8.10 Test implementovaných algoritmů 79
8.11 FPGA procesor genetického algoritmu 84
8.12 ASGG generátor 84

SEZNAM TABULEK

1.1 Umělá inteligence (AI) – přehled 24
1.2 Porovnání algoritmů výpočetní inteligence 25
1.3 Přehled modelů 26
1.4 Příklady systémů detekce a analýzy dat 29
2.1 Genetický algoritmus – pojmy 38
2.2 Počáteční populace a její ohodnocení 40
2.3 Výpočet ruletového kola 40
2.4 Přehled typů křížení v GA v závislosti na formě jedince 43
A.1 NetFlow verze 5 – význam položek datagramu 104
B.1 Částečný výčet typů evolučních algoritmů 105

SEZNAM PROGRAMOVÉHO KÓDU

1.1 Jarník-Primův algoritmus 16
2.1 Základní princip EA 35
2.2 Algoritmus DE 37
2.3 Mutace populace 43
6.1 Metoda selekce údajů z databáze 55
6.2 Metoda Kaplan-Meier analýzy 57
8.1 Metoda selekce údajů z databáze 68
8.2 Metoda časové analýzy v GDP 69
8.3 K-Means: přiřazení centroidu a členů 73
8.4 K-Means: Euklidovská vzdálenost 74
8.5 K-Means: konvergence 74
8.6 Kriteriální funkce GA shluků 75
8.7 Běh algoritmu GA shlukové analýzy 77
8.8 Křížení populace v GDP 78
8.9 Mutace populace v GDP 78

ÚVOD

Předkládaná dizertační práce se věnuje tématu z oblasti konvergovaných sítí, zaměřuje se na problematiku síťové tomografie, respektive monitoringu a zpracování komunikačních dat s využitím evolučních algoritmů. Vlastní výzkum volně navazuje na projekty LO1401 a FEKT-S-14-2352 a na vlastní řešené projekty „Detekce bezpečnostních hrozeb na aktivních prvcích kritických infrastruktur“, VI20172019072 a „Redukce bezpečnostních hrozeb v optických sítích“, VI20172020110 s návazností na vyhlášené soutěže MV ČR.

Samotný výzkum v rámci dizertační práce byl převážně realizován za pomocí laboratorních síťových zařízení Cisco, jmenovitě Cisco 800, Cisco 1841, Cisco 2960, Cisco 2691. Proto i část výsledků je přizpůsobena pro tato síťová zařízení. Dále bylo využito protokolu NetFlow, GNS3 simulátoru ve verzi 1.3.11 a zdrojů z Cisco Devnet [1]. Jako hlavní programovací jazyk je zvolen Python verze 3.4 s prostředím pro vývoj PyCharm Professional 2016.2.

Cíl práce

Hlavním cílem dizertační práce je návrh nové metody detekčních mechanizmů v oblasti konvergovaných sítí. Cílem je využít současných poznatků možností vyhodnocení chování a výskytu anomálií provozu v takovýchto sítích a navrhnout řešení nová. Podnětem pro práci samotnou jsou existující problémy spojené s behaviorální analýzou dat v konvergovaných sítích.

Součástí vytyčených cílů je návrh a implementace algoritmu pro analýzu a detekci dat v konvergovaných sítích. Tento algoritmus, či skupinu algoritmů modelovat a otestovat v některém z programovacích či skriptovacích jazyků.

Teoretický problém: Výzkum nových mechanizmů detekce dat a jejich vzájemných souvislostí a možnosti predikce jejich chování v konvergovaných sítích s využitím evolučních algoritmů a metod tomografie sítí.

Aplikovaný problém: Vyhodnocení stávajících síťových mechanizmů k detekci dat a vytvoření mechanizmů nových.

Způsoby řešení definovaných problémů: Při rozboru daného tématu je uvažována kvantitativní orientace, a to vzhledem k charakteru zkoumaných síťových jevů. Podstata je ve výsledku chování zkoumaného objektu. Dále jsou také zvoleny metody jak teoretické, tak i empirické. Z teoretických metod se jedná konkrétně o analýzu a modelování. Z empirických metod se jedná o měření a experiment.

Výsledky a výstupy v bodech

1. Analýza současného stavu algoritmů a prostředků k detekci provozu.
2. Návrh nové metody detekce anomálií provozu.
3. Vývoj algoritmu, či skupiny algoritmů vycházejících z algoritmů evolučních.
4. Model, který bude ověřovat funkčnost těchto algoritmů.
5. Publikační činnost.

Organizace práce

Úvodní kapitola obsahuje přehled organizace práce, cíle práce a její výsledky. V kapitole první se práce věnuje rozboru jednotlivých témat vztahujících se k dané problematice a současnému stavu poznání (*State of Art*). Tato problematika obsahuje: popis současných konvergovaných sítí, definice pojmu tomografie síťového provozu, rozbor metod používaných k detekci anomálií a detekcí provozu, specifikace síťových zařízení a protokolů ke sledování provozu.

Kapitola druhá se věnuje rozboru základních technik detekcí anomálií síťového provozu. Jsou zde rozebrány jednotlivé principy a jejich matematický základ. Kapitola třetí obsahuje teoretický úvod evolučních algoritmů. Je zde probrána příslušná teorie, se zaměřením na genetický algoritmus. Jsou zde popsány operátory genetických algoritmů selekce, křížení a mutace.

V následujících dvou kapitolách jsou uvedeny jednotlivé stěžejní publikace k daným tématům a rozebrána perspektiva řešení výzkumného záměru. V kapitole páté je uveden rozbor a testování vývojových a simulačních programů za účelem zvolení vhodného vývojového prostředí pro studium a vývoj. Je zde proveden srovnávací výkonový test programu napsaného v Python a OMNeT++ simulující síťový provoz.

Kapitola šestá se věnuje testování hypotézy analýzy přežití. Je zde ověřena a testována možnost definovat provoz a jeho životní cyklus v závislosti na podobnosti. V kapitole sedmé jsou uvedeny tři ilustrace problémů spojených s evolučními algoritmy, a to maximalizace funkce, výkonnost a výpočet NP (*Nondeterministic Polynomial*).

Kapitola osmá se zabývá návrhem algoritmů k detekci provozu a funkčním modelem síťové sondy, kde jsou algoritmy implementovány. V této kapitole jsou také uvedeny výsledky ověřující funkčnost daného řešení.

Praktické příklady problematiky genetického algoritmu a výsledné kódy ke každé z problematik jsou součástí přílohy dizertační práce. Příloha obsahuje autorovy zdrojové kódy rozdělených dle témat.

1 PŘEHLED SOUČASNÉHO STAVU

Tato kapitola shrnuje současný stav řešení daného tématu, rozděleného do jednotlivých podkapitol, dále pak i studovanou literaturu k problematice.

1.1 Konvergované sítě

Současné konvergované sítě umožňují vzájemnou kooperaci na výkonové a funkční úrovni. Moderní koncept sítí poskytuje možnost nasazení moderních metod k řízení provozu, jejich konfiguraci a zabezpečení. Pojem konvergence sítí se také mění s nástupem softwarově definovaných sítí. Tento vývojový směr je označován zkratkou SDN (*Software Defined Network*). Cílem SDN je oddělení kontrolních funkcí síťových zařízení používajících aplikační programové rozhraní API od samotných dat. Jak uvádí autor Stallings [2], jednou z iniciativ společnosti Cisco [3] v této oblasti je standardizace technologie „OpenFlow“. Vlastní kompletní řešení SDN byla nasazena i společností Google Inc.

Sbližování technologií a zpracování dat jde v ruku v ruce s virtualizací. Koncepce pevně definované hierarchické infrastruktury sítí neposkytuje integraci všem síťovým zařízením. Nahrazuje ji infrastruktura, kde se hovoří o slučování technologií, někdy i celých technologických firem. V této oblasti je významným inovátorem společnost Cisco [3], která přichází s řešením ACI (*Application Centric Infrastructure*).

V popředí jsou také standardizace nových síťových protokolů. Jedná se zejména o aktivity mezinárodní standardizační organizace NGN-GSI (*Next Generation Networks Global Standards Initiative*). Od nasazení internetového protokolu pro komunikaci IPv6 [L1] je také stále více diskutováno o technologiích s názvem IoT (*Internet of Things*) a o trendu BYOD (*Bring Your Own Device*). Se zvyšujícím se počtem hustoty osobních komunikačních zařízení je možno vidět příchod IoT v D2D (*Direct Device to Device*) komunikaci[1].

Základní myšlenky konvergence jsou promítány i do oblasti mobilních a bezdrátových sítí. Za současný vývojový trend jsou považovány sítě souhrnně označované názvem technologie páté generace (*5G*), která doplňuje generaci mobilních sítí LTE a všeobecně bezdrátových sítí. Jednotlivými výzvami [4] a vývojovými trendy v této mobilní technologii 5G jsou oblasti komunikace mezi zařízeními (*Machine to Machine Communication*), zkráceně M2MC, D2D, oblast přístupu, vývoj nové archi-

[1] D2D – je typ komunikace mezi dvěma a více síťovými zařízeními, která využívají alternativního připojení při nedostupnosti primárního spojení, například při ztrátě signálu v LTE (*Long Term Evolution*) síti.

tektury rádiového přístupu, samoorganizujících se sítí, bezpečnosti, monitoringu a virtualizace jádra sítě.

Otázka bezpečnosti konvergovaných sítí nabývá stále na významu. Pravidelné roční zprávy předních firem jako jsou Cisco, Prolexic, Acamai, Radware [5, 6] vykazují nárůst bezpečnostních incidentů, a i jejich vysoký podíl na celkovém provozu s nárůstem až 114 % [7]. Přináší důkazy o výskytech omezení přenosového pásma, saturace konektivity útokem generujícím data nad 158 Gb/s. V prvním čtvrtletí roku 2014 se podařilo firmě Prolexic úspěšně eliminovat rozsáhlý DDoS (*Distributed Denial of Service*) útok vedený proti jejich firemním zákazníkům v centrech ve Frankfurtu a Londýně. Tento celkový nevyžádaný provoz dosahoval kapacitní špičky až 200 Gb/s. K tomuto útoku byly využity protokoly NTPv2, DNS a HTTP (POST). V roce 2016 byl v USA proveden masivní útok typu DDoS vedený botnetovou sítí Mirai, který generoval provoz o velikosti 1,2 Tb/s [8].

Podkladem k výzkumu mohou být i výzvy CSIRT (*Computer Security Incident Response Team*) týmů. Příkladem je provedená vlastní analýza [A2] „varování ISP (*Internet Service Provider*) zákazníků xDSL proti potencionální síťové hrozbě“ [9]. V důsledku se i přes nasazení bezpečnostních technologií nepodařilo jednoznačně určit vektor útoku[2].

Nevyžádaný provoz výrazně zasahuje do poskytovaných služeb zákazníkům. V zásadě již nezáleží na použitých nástrojích, ale na způsobu provedení síťových útoků. Současným trendem je použití hybridního přístupu, kdy útočníci typicky mixují několik operací dohromady k vytvoření mnoha vektorů útoku. Během těchto operací jsou měněny vektory útoku a signatury protokolů za účelem oklamání automatizovaných mitigačních zařízení. Nejefektivnější síťové útoky využívají předem získaných znalostí z otisků (*footprint*) jednotlivých mitigačních zařízení, která se poté mohou stát neúčinná.

Kupříkladu dle CVE-2007-0540 [10] byla v aplikaci WordPress zneužita tzv. „pingback“ automatická funkce určená k informování administrátorů. Byl přesměrován požadavek metody *GET* a ve velkém rozsahu záplavově šířen. Tento typ útoku dle dostupných statistik firmy Akamai [7] generoval v roce 2014 v průměru až 28 Mb/s na jednu zneužitou aplikaci.

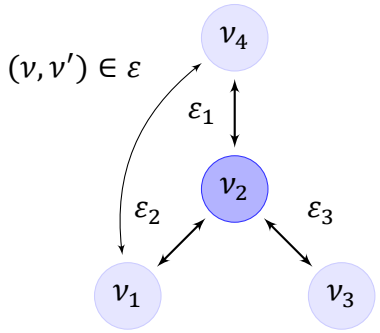
Stále více jsou v popředí zájmů útočníků mobilní klienti a chytrá zařízení využívající konvergovaných služeb hlasu a dat. Publikované závěry uvedených firem poukazují na nutnost vývoje bezpečnostních zařízení, která boudou schopna v reálném čase reagovat na útoky a anomálie síťového provozu. Nezbytné je začlenění těchto zařízení v SDN sítích. Vývoj v konvergovaných sítích přináší nová bezpečnostní rizika, jako jsou útoky „Denial of Energy“ nebo „Denial of Company Reachability“ [11].

[2] *Vektor útoku* – je cesta nebo kombinace prostředků, kterými může útočník získat přístup k síťovému zařízení.

1.2 Tomografie síťového provozu

Pojem „tomografie“ není zcela nový, ale je převzat z anglického významu pro výzkum vlastností chování sítě na základě jejího vnějšího a vnitřního pozorování. První výrazné zmínky o síťové tomografii (*Network Tomography*) byly uvedeny autorem Vivardi [12]. V tomto díle bylo snahou zachytit vztah mezi maticí výchozího provozu (*Origin Destination Matrix*) a počtem propojení jednotlivých uzlů sítě. Síťová tomografie je disciplína, která studuje interní chování a charakteristiku datového provozu a sítě pomocí externích zařízení (koncových bodů).

Tato zařízení mohou být reprezentována specializovanými hardwarovými sondami, o IDS systémech bude pojednáno v kapitole 1.3, ale také jednotlivými prvky, jako jsou směrovače, počítače, mobilní zařízení a technologie IoT. Všechna tato zařízení mohou poskytovat data pro účely analýzy. Základní princip síťové tomografie je uveden na obrázku 1.1.



Obr. 1.1: Reprezentace síťové tomografie pomocí grafu uzlů

Je zde využit teorém grafu. Nechť je graf $G_i = (v, \varepsilon)$, kde jednotlivé uzly v reprezentují síťová zařízení s vrcholy $v = \{1, 2, 3 \ldots |v|\}$ a ε reprezentujícími linky mezi nimi, tedy hrany $\varepsilon \subseteq v \times v$. Potom $(v, v') \in \varepsilon$ značí přímou cestu P_G a $\delta_G(v, v')$ nejkratší cestu P_{δ_G} z v do v'. Modře znázorněné body v zde tedy reprezentují jednotlivá síťová zařízení a spojení mezi nimi jsou reprezentována spojnicemi označená symbolem ε.

Síťová tomografie proklamuje, že je možné efektivně mapovat směr datového provozu, její kapacitu, kvalitu a prováděné útoky pomocí informací získaných z dat, ať jsou pasivně ukládána nebo aktivně zkoumána v reálném čase. Základními technikami síťové tomografie jsou zkoumány vlastnosti sítě, jako je zpoždění a ztrátovost dat. Inovované techniky síťové tomografie například zkoumají chování dat v reálném čase nebo také typizaci dat a jejich shluků v čase.

Z uvedeného příkladu na obrázku 1.1 je možno odvodit základní matematický vztah pro zjištění jednotlivých hodnot zpoždění každého síťového propoje. Tento výpočet je uveden ve vztahu (1.1). Z vektoru celkového zpoždění $\boldsymbol{v}$ a matice spojení

M lze odvodit jednotlivé hodnoty zpoždění jednotlivých spojnic ve vektoru $\boldsymbol{\varepsilon}$. Hodnoty v tomto vektoru potom představují výsledné řešení. Z uvedených základních matematických principů vyplývá, že popsané metody síťové tomografie je možné využít v širším měřítku, a to zejména pro analýzu globálního provozu konvergovaných sítí za předpokladu užší spolupráce jednotlivých provozovatelů sítě.

$$\boldsymbol{\nu} = \mathbf{M}\boldsymbol{\varepsilon} \;\; ; \qquad \begin{bmatrix} \nu_1 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \tag{1.1}$$

Výsledky síťové tomografie mohou jednoznačně přispět k vytváření pravděpodobnostních modelů chování datového toku, objevování shluků dat a bezpečnostní analýze [14]. Techniky síťové tomografie a její principy jsou využívány komunitou RIPE NNC [15] a jejími nástroji RIPE Atlas, využívající síťové sondy k měření globální síťové konektivity a dostupnosti v reálném čase.

1.2.1 Algoritmy grafu

Základními algoritmy používanými pro problematiku grafu jsou Jarník-Primův algoritmus, Borůvkův algorimus, Kruskalův algoritmus, Floydův-Warshallův algoritmus, Tarjanův algoritmus, nepostradatelný most, Edmondsův algoritmus, Dijkstrův algoritmus, Bellman-Fordův algoritmus a jiné. Je možné nalézt více modifikací těchto algoritmů řešících konkrétní problematiku. Principy těchto uvedených algoritmů lze nalézt v [16].

Pro vhled do problematiky je uveden základní Jarník-Primův algoritmus. Tento algoritmus slouží k určení minimální kostry grafu takové, že celkový součet vah jeho hran je minimální. Uvažujme souvislý graf G s vrcholy v a hranami ε, potom tento algoritmus hledá minimální kostru T_k (*Minimum Spanning Tree (MST)*) G.

```
Inicializuj ε = ϕ a prostor T_k = {v} pro náhodné v
Opakuj dokud T obsahuje vrcholy v:
        nechť je ε hrana s nejmenším ohodnocením a jedním koncovým bodem v T_k
        přiřaď další koncový bod do T_k
        přiřaď další hranu ε do T_k
```

Programový kód 1.1: Jarník-Primův algoritmus

Při použití matice sousednosti představuje T_k soubor vrcholů v v aktuálním stromu a pro každý vrchol, který není v T_k udržuje vrchol v T_k, který je nejblíže. Pro přidání dalšího vrcholu do T_k využije ohodnocení min $dist[w]$. Před jeho přidáním je provedena operace – pro každého souseda (w, v) prozatím nepřidaného do T_k , pokud je w blíže vrcholu v v T_k, aktualizuj $dist[w]$. Náročnost takového algoritmu je $O(V2)$.

Při použití binární haldy $O(E \log V)$. Zajímavé je srovnání, který z použitých algoritmů má rychlejší zpracování. Záleží na počtu vrcholů. Od řádu 100 000 vrcholů je binární halda 500× rychlejší než výše uvedený algoritmus. Klasický Jarník-Primův algoritmus je vhodný pro husté grafy, které obsahují „mnoho“ hran vzhledem k počtu uzlů.

1.3 Základní techniky detekcí anomálií v síťovém provozu

Anomálie[3] síťového provozu mohou být rozlišovány v závislosti na jejich výskytu a původu. Jak uvádí autor Eduardo B. Fernandez [17], základními technikami používanými v praxi jsou použití algoritmů výpočetní inteligence, verifikace protokolů či statistického modelování. Detekci anomálií je možné dle prostudované literatury dále rozdělit podle použité metodologie, a to na detekce anomálií založené na bázi:

- Signatur, verifikace protokolů.
- Signálové analýzy – statistických metod.
- Inteligence.
- Kombinace výše uvedených metod.

1.3.1 Techniky verifikace protokolů

Metody detekce anomálií používající techniky verifikace protokolů zakládají na takovém přístupu, že sledují neregulérní vzhled a obsah hlaviček jednotlivých síťových protokolů, které nejsou „správně“ předány a zpracovány aplikačními systémy. Verifikace obsahuje zejména tři stavy. Úspěch, nezdar nebo chybu. Úspěchem se rozumí, že všechny dané body verifikace se shodují. Oproti tomu nezdar znamená neočekávanou odezvu. Chyba je potom odezva, kterou nebylo možné analyzovat.

Mezi základní verifikační metody patří ověřování sekvenčních čísel na TCP (*Transmission Control Protocol*) segmentu, verifikace IP (*Internet Protocol*) kontrolního součtu, kontrola navazování TCP spojení (*handshake*), kontrola fragmentace, verifikace ACK, zjišťování přesměrování provozu a jiné. Představují dnes již nezbytnou výbavu IDS systémů a jsou průběžně aktualizovány.

[3] *Anomálie* – představuje odchylku od nějakého pravidla nebo normálního jevu. V informatice se anomálie označuje jako odlehlá hodnota a data, která se výrazně liší od ostatních či referenčních dat, se v angličtině obecně nazývají „outlier“.

1.3.2 Techniky statistického modelování a analýzy

Techniky statistického modelování a analýzy jsou velice hojně používány a jedná se o velice rozsáhlý obor překračující možnosti této práce. V následující části je proto uveden jen výčet statistických metod a výpočtů z prostudované literatury, které byly nejčastěji použity k detekci anomálií v IP sítích.

Pro detekci anomálií se používají buďto multivariantní (vícerozměrné) modely nebo modely založené na dostupných statistikách jako je entropie dat, komprese či měření průměru a odchylek předem definovaných profilů. Ve statistických metodách je také často používána shluková analýza, kde jednotlivé shluky dat představují podobné aktivity chování a vzhledu dat. V této metodě se používají i předlohy obvyklého chování jednotlivých uživatelů sítě. Základními principy měření a analýzy anomálií jsou jednoduchý průměr, měření entropie dat a porovnávání podobnosti.

Jednoduchý průměr

Ve statistice existuje mnoho přístupů, jak získat charakteristiku zkoumaných dat [18]. Jedním z nejjednodušších způsobů je vypočítat klouzavý průměr. Jednoduchý a vážený klouzavý průměr představují základní používané metody v analýze anomálií, a to nejenom v telekomunikacích, ale i v jiných odvětvích jako jsou například finanční trhy a sociální sféra.

V případě jednoduchého klouzavého průměru SMA (*Simple Moving Average*) se jedná o nevážený průměr n čísel v časové řadě. Například, pokud je uvažováno číslo d_t v čase t, a potom d_{t+n-1} bude číslo v čase $t + n - 1$, potom jednoduchý klouzavý průměr lze zapsat jako (1.2):

$$SMA = \frac{\sum_{i=0}^{n-1} d_{t+i}}{n} = \frac{1}{n} \sum_{i=0}^{n-1} d_{t+i} \tag{1.2}$$

Jednoduchý klouzavý průměr zachycuje průměrnou změnu hodnot v jednotlivých časových oknech, či slotech, ale ztrácí informace o jejich výkyvech a propadech.

Entropie dat

Využití pouze klouzavého průměru není dostatečná natolik, aby odrážela jednotlivé aspekty anomálií [18]. Entropie (stav neuspořádanosti) v informační technologii je také nazývána jako Shannonova entropie po autorovi Claude Elwood Shannonovi. Samotný pojem však zavedl fyzik Rudolf Julius Emanuel Clausius v roce 1865.

Jde o výpočet informačního množství nějakého celého jevu. Pokud je předpokládáno, že zkoumaný jev S má n realizací s pravděpodobnostmi P, pak střední hodnota vlastních informací všech realizací jevů je definována vztahem (1.3). Pokud je takovýmto jevem výskyt prvku signálu, potom je jeho jednotkou [Sh/prvek].

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i) \qquad [Sh/bit]\,, \tag{1.3}$$

kde S představuje systém s konečným počtem možných stavů $S \in \{s_1, s_2, \dots, s_n\}$ a $P(s_i)$ je pravděpodobnostní distribucí stavu S.

Existuje nesčetně mnoho publikací s ohledem na možnosti detekce anomálií pomocí modifikovaných výpočtů entropie provozu. Dále je uveden výčet některých z těchto publikací. Detekce anomálií provozu založené na takovémto přístupu sledují entropii vybraného objektu, například anomálie DNS či HTTP protokolu. Porovnávají současnou hodnotu entropie s referenčními hodnotami. Při výkyvu či překročení stanovených hraničních hodnot je detekována anomálie.

Autoři Przemysław Berezinski a kolektiv [19] porovnávali jednotlivé druhy přístupu detekcí malware[4] pomocí entropie ve vzorcích dat sítě. Porovnávali schopnosti detekce při použití entropie podle Shannona, Rényi [20] a Tsallise. Došli k závěru, že detekce moderních botnetových sítí [5] na základě entropie je proveditelná. Nejlepší výsledky podávaly výpočty dle Rényi a Tsallise.

Autoři Mobin Javed a kolektiv [21] ve svém článku provádí základní výzkum v oblasti efektivity entropie vzhledem k možnostem detektorů. Větší efektivitu prokazovala entropie relativní. Ta je dána poměrem entropie a její maximální hodnoty.

Koncept vzdálenosti

Mnoho multivariantních technik (vícerozměrných analýz) a modelů aplikovaných na detekci anomálií jsou založeny na konceptu vzdálenosti. Nejznámější metrikou je Euklidovská vzdálenost. Jako taková je hojně používána pro měření spojitostí či similarity. Jsou–li brány v úvahu dva vektory $\mathbf{x} = (x_1, x_2, \dots, x_n)$ a $\mathbf{y} = (y_1, y_2, \dots, y_n)$ jako dvou dimenzionální zjištění hodnot, potom Euklidovská vzdálenost mezi $\mathbf{x}$ a $\mathbf{y}$ je definována dle vztahu (1.4).

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})} \tag{1.4}$$

Protože každá hodnota vlastního vektoru přispívá co do výpočtu Euklidovské vzdálenosti, mohou se výsledky výrazně lišit i při malé změně těchto hodnot. Hraje zde

[4] *Malware* – představuje počítačový kód určený k vniknutí do počítačového systému.

[5] *Botnet* – je kombinace slov robot a síť. Jedná se o velmi obtížně identifikovatelný provoz, většinou určený pro šíření škodlivého kódu a provedení DDoS útoku.

roli i dominance jedné sady hodnot vůči druhé sadě hodnot. Proto se variabilita dá zanést přímo do výpočtu. Jednou z nejznámějších metrik se zavedenou variabilitou je Mahalanobisova vzdálenost (1.5), kde **V** představuje váženou disperzní matici - kovariancí parametrů.

$$d^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{V}^{-1} (\mathbf{x} - \mathbf{y}) \tag{1.5}$$

Matice kovariancí parametrů je zobecnění pojmu rozptylu pro náhodné vektory. Pokud je tato matice identickou maticí, je Mahalanobisova vzdálenost redukována na Euklidovskou vzdálenost. Pokud je tato matice diagonální, pak se jedná o normalizovanou Euklidovskou vzdálenost.

Dalším typem měření vzdálenosti použité v detekci anomálií je Canberrská metrika, která je určena pouze pro nezáporné proměnné, rovnice (1.6).

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{|\mathbf{x} - \mathbf{y}|}{(\mathbf{x} + \mathbf{y})} \tag{1.6}$$

PCA – analýza hlavních komponent

Analýza hlavních komponent PCA (*Principal Components Analysis*) [22] představuje cestu, jak identifikovat vzory v sadě dat s vysokou dimenzí a zvýraznit jejich odlišnosti a podobnosti. Slouží také ke snížení dimenze samotných dat. Jelikož jsou nalezeny tyto vzory, je možné provést kompresi. PCA představuje transformaci do souřadnic, které mapují sadu n dimenzionálních datových bodů na n nových, nekorelovaných proměnných, které se nazývají hlavní komponenty.

Tyto komponenty mají následující vlastnosti. Charakteristikou každé komponenty je její rozptyl. Všechny komponenty jsou seřazeny podle tohoto rozptylu od největšího k nejmenšímu. První hlavní komponenta popisuje největší část rozptylu původních dat, druhá hlavní komponenta popisuje největší část rozptylu neobsaženého v první hlavní komponentě a je kolmá na první komponentu. Třetí hlavní komponenta popisuje největší část rozptylu, který není obsažen v předešlých a je na ně kolmá, atd. Zjednodušený postup je následující:

- Příprava dat.
- Získání vlastních vektorů a vlastních hodnot matice z matice kovariancí parametrů, korelační matice nebo provedení vektorové dekompozice (závisí na řešeném případu).
- Setřídění sestupně vlastních hodnot matice a zvolení vlastních vektorů, které korespondují největším vlastním hodnotám.
- Vytvoření transformační matice **T** ze zvolených vlastních vektorů.

– Transformace původních dat, které tvoří matici **X** pomocí nově vytvořené matice vlastních vektorů.

Klasickým přístupem je provedení dekompozice na matici **V** kovariancí parametrů o rozměrech $d \times d$, kde každá položka matice **V** reprezentuje kovarianci mezi dvěma hodnotami x_{ij}, x_{ik}, která je vypočítána následovně dle (1.7).

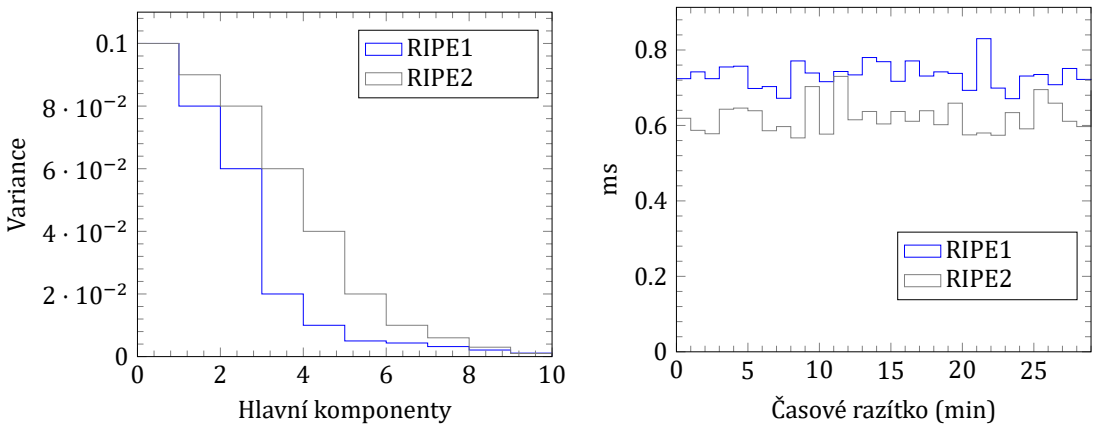
$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \tag{1.7}$$

Potom je možné sumarizovat výpočet kovarianční matice, jak je uvedeno v (1.8).

$$\mathbf{V} = \frac{1}{n-1} \left((\mathbf{X} - \bar{\mathbf{x}})^T (\mathbf{X} - \bar{\mathbf{x}}) \right), \tag{1.8}$$

kde $\bar{\mathbf{x}}$ představuje průměrný vektor, což je d dimenzionální vektor, kde každá hodnota reprezentuje průměrnou hodnotu ze všech hodnot jedné měřené položky. Výsledně je možné provést transformaci $\mathbf{Y} = \mathbf{X} \times \mathbf{T}$.

Provozní zátěž síťových linek ma nízkou efektivitu dimenzí [23], pomocí PCA analýzy lze tedy efektivně detekovat anomálie. Na obrázku 1.2 je zobrazena variance dvou linek, která byla sledována síťovou sondou v laboratoři transportních sítí [24].



Obr. 1.2: PCA analýza variance latence provozu ze síťových sond

Provozní zátěž zde nahrazuje hodnota obousměrného zpoždění RTT (*Round–Trip Time*)[6]. Je zde možné vidět, že první tři hlavní komponenty nesou nejvíce variance. Tato variance byla vypočítána programem Python dle postupu v [25].

[6] *RTT* – obousměrné zpoždění, je časová hodnota, kterou zabere signálu pro přenos z jedné stanice na druhou, až po návrat potvrzení tohoto přenosu na první stanici. Původní algoritmus byl nahrazen Jacobson/Karels algoritmem, který bere v potaz směrodatnou odchylku.

Vícerozměrné metody pro analýzu a klasifikaci dat

[7]Vícerozměrná data jsou definována více než jednou proměnnou či charakteristikami m v datové sadě n. Objekty jsou popisovány ve formě matice $\mathbf{X}$ o rozměrech $m \times n$ (1.9).

$$\mathbf{X}_{(m,n)} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix} \tag{1.9}$$

K jejich zobrazení jsou používány maticové, krabicové či ikonové grafy. V jejich řešení jsou známy problémy chybějících dat či dvou nul. Pro výpočty podobností a vzdáleností ve vícerozměrném prostoru jsou používány zmíněné koncepty vzdálenosti z předchozí podkapitoly. Do této kategorie patří dále Hammingova metrika vzdálenosti. Ta je dána součtem všech prvků kontingenční matice. Mezi nedeterministické metriky určení vzdálenosti mezi dvěma vektory patří metrika nejbližšího souseda, metrika nejbližších sousedů, metoda nejvzdálenějšího souseda.

Pro testování shody mezi dvěma vícerozměrnými soubory lze určit pomocí korelace datových matic. Nejprve je spočítána matice vzdáleností a danou korelaci mezi těmito maticemi lze spočítat pomocí Mantelova testu (1.10). Nulová hypotéza testu předpokládá, že vzdálenost mezi objekty matice nekorelují se vzdáleností matic. Základní statistikou $S_{\mathbf{M}}$ je tedy suma všech prvků matice mimo prvků na diagonále, viz rovnice (1.10).

$$S_{\mathbf{M}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} y_{ij} \tag{1.10}$$

Ke třídění získaných dat z vícerozměrných pozorování jsou využívány techniky shlukování. Data jsou setříděna tak, aby se rozdíl hodnot dat členů skupiny blížil k nule. Shluková analýza se zabývá právě tvorbou takovýchto homogenních celků. Je snížen počet dimenzí dat a jedna proměnná vyjadřuje příslušnost datové jednotky ve shluku.

Postup shlukování je možné popsat obecně tak, že je k dispozici datová matice $\mathbf{X}_{(m,n)}$, kde m je počet objektů a n je počet proměnných. Počet shluků je značen k. Jedná se o rozklad množiny m objektů v závislosti na hodnotách n do k shluků. V potaz jsou brány pouze rozklady s disjunktními shluky. Jeden objekt musí patřit pouze jednomu shluku S_k. Je vypočtena vzdálenost pro všechny objekty. Z tohoto výpočtu vznikne symetrická čtvercová matice zvaná asociační matice.

[7]Uvedená problematika je velice rozsáhlá, jsou proto uvedeny základní příklady, které jsou použity následně v řešení vlastní práce. Je čerpáno z literatury matematické biologie [26], a také z absolvovaných kurzů umělé inteligence.

Metody shlukování jsou rozdělovány na hierarchické nebo nehierarchické. Záleží na složení a struktuře vstupních dat. Nehierarchické shlukování je vhodné pro velké objemy dat, mezi něž patří metoda K-průměrů, metoda X-průměrů a metoda K-medoidů. Následně je uveden princip metody K-průměrů. Tato metoda patří mezi výše uvedenými za jejich základ.

1. Data jsou náhodně rozdělena do k shluků.
2. Je určeno k centroidů[8] c_k pomocí konceptu průměru vzdálenosti ve shluku.
3. Je hodnocen každý objekt shluku a jeho vzdálenost k centroidu. Pokud má blíže k jinému, je přemístěn a centroidy jsou opět přepočítány tak, že je vypočítán nový průměr ze všech prvků shluku.
4. Je opakován předchozí bod do doby, kdy žádný z prvků již není možné přemístit.

Matematicky je možné vyjádřit vztah k shluků S_k a k centroidů c_k minimalizací S_k a c_k dle následujícího vztahu (1.11).

$$\sum_{k=1}^{k} \sum_{x_n \in S_k} \|x_n - c_k\|^2 \tag{1.11}$$

Problém minimalizace představuje těžkou úlohu řešení. Nejznámějším řešením je pomocí Lloydova algoritmu. Jakmile jsou známy centroidy, jsou prvky přiřazeny dle koncepce vzdálenosti dle následujícího vztahu (1.12).

$$S_k = \{x_n : \|x_n - c_k\| \leq \forall \|x_n - c_k\|\}, \qquad c_k = \frac{1}{S_k} \sum_{x_n \in S_k} x_n \tag{1.12}$$

Existuje několik modifikací tohoto algoritmu. Mezi nevýhody patří pevná definice k shluků a využití výpočtu Euklidovské vzdálenosti, které je náchylné na vzdálené objekty. Pro validaci počtu k shluků existují opět metody jejich validace, jako je validační metoda siluety či Daviesův-Bouldinův validační index DB (*Davies-Bouldin Validity Index*). Daviesův-Bouldinův validační index vychází z podílu sumy rozložení uvnitř shluku a rozložení mezi shluky. Tento index je získán ze vzorce (1.13).

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left\{ \frac{S_n(Q_i) + S_n(Q_j)}{S_n(Q_i, Q_j)} \right\}, \tag{1.13}$$

kde n je počet shluků, $S_n(Q_i)$ je průměrná vzdálenost uvnitř shluku od jeho středu a $S_n(Q_i, Q_j)$ je vzdálenost mezi jednotlivými shluky reprezentované centroidy.

[8]Centroid je střed shluku. Jedná se o vektor obsahující průměry proměnných pozorovaných ve shluku.

1.3.3 Techniky výpočetní inteligence

V oblasti počítačové bezpečnosti jsou používány algoritmy vycházející z biologicky inspirovaných metod. Patří sem zejména neuronové sítě, výpočty pomocí evolučních algoritmů a umělé imunitní systémy AIS (*Artificial Immune Systems*), tabulka 1.1.

Tab. 1.1: Umělá inteligence (AI) – přehled

Přírodní prototyp	Biologická úroveň	Model
Řeč	Levá hemisféra mozku	Formální logika
Mozková nervová síť	Buňky	Neuronové sítě
Biologické buňky	Buňky	Celulární automat CA
Molekuly proteinů	Molekuly	Umělý imunitní systém AIS
Genetický kód	Molekuly	Genetické algoritmy GA

Umělé imunitní systémy

Umělé imunitní systémy AIS jsou inspirovány imunitním systémem lidského těla. Jedná se o decentralizované, adaptabilní a chybově tolerantní systémy. Existuje celá řada modelů AIS použitých v rozpoznávání a detekci anomálií. Používají zejména zjednodušené modely různých imunologických procesů a technik [27, 28, 29]. Tyto mechanizmy jsou používány zejména v systémech IDS, v těžení dat a problému klastrování dat.

Pro detekci anomálií jej použil Dasgupta [28] (1996), pro rozpoznávání vzorů dat autoři Forest (1993), Gibert (1994). K těžení dat byla tato metoda použita autory Huntem (1996) a Timmisem [27] (2001–2002). Třemi základními popsanými technikami (mechanizmy) jsou: teorie imunitní sítě, záporný selekční mechanizmus a princip klonální selekce. Dalšími jsou Bone-marrow model, afinitní[9] funkce a somatická hypermutace. Porovnání jednotlivých algoritmů je uveden v tabulce 1.2.

Využití umělých imunitních systémů se jeví velice perspektivní ve všech oblastech bezpečnosti sítí. Tato práce se však zabývá využitím principů algoritmů evolučních, detailnější přehled o principech AIS a algoritmy lze například čerpat z [30].

Genetické algoritmy

Systémy používající modifikované genetické algoritmy vyhovují současným požadavkům v oblasti rozhodovacího procesu a přistupují k modelování chování síťových protokolů a aplikací tak, že zavádí rekurzi odlišných stavů provozu, jako je „normální“ chování, chybový stav a stav útoku. Detailněji vysvětleno včetně zde uváděných pojmů v kapitole 2.1.

[9] *Afinita* – síla vazby ligandu (molekuly) ke svému receptoru.

Tab. 1.2: Porovnání algoritmů výpočetní inteligence

Vlastnost	**GA – optimalizace**	**NN – klasifikace**	**AIS**
Komponenta	Chromozóm	Umělé neurony	Řetězce atributů
Umístnění komponenty	Dynamické	Předdefinované	Dynamické
Struktura	Diskrétní komponenty	Síťové komponenty	Kombinace
Znalostní báze	Diskrétní komponenty	Síťové komponenty	Kombinace
Princip	Evoluce	Učení	Kombinace
Interakce mezi komponentami	Křížení	Síťová spojení	Rozpoznávání a síťová spojení
Interakce s prostředím	Ohodnocující funkce	Externí stimulant	Objektivní funkce a rozpoznávání

Pracují většinou ve dvou krocích. První krok algoritmu zahrnuje kódování vstupní populace. Ve druhém kroku zahrnují nalezení fitness funkce k testování jednotlivých individuí populace v závislosti na určitých evolučních kritériích. V procesu učení každá sekvence chování síťového uzlu formuje základní gen. Fitness funkce je kalkulována pro celou kolekci genů. Pokud není v aktuální generaci nalezen gen s požadovanou fitness funkcí, je vyvinut nový set genů pomocí křížení a mutace. Tento proces pokračuje do chvíle, kdy je nalezen odpovídající set genů s požadovanou hodnotou fitness funkce. Definiční proces, který používají systémy k detekci bezpečnostních anomálií datového provozu zahrnuje definování vektorů pro události cyklu genetických algoritmů a testovací metody, zda takovýto vektor události indikuje útok, proniknutí, či nikoliv.

Mnohé genetické algoritmy byly použity pro vylepšení stávájících metod detekce, či predikce anomálií. Autoři Divya Somvanshi a R.D.S. Yadava [31] použili GA (*Genetic Algorithm*) v PCA analýze pro extrakci komponent. Autor Wilson Rivera-Gallego [32] využívá genetického algoritmu pro výpočet matic Euklidovské vzdálenosti.

1.3.4 Síťová tomografie v roli detekcí anomálií

Síťová anomografie (*Network Anomography*) – autorů Yin Zhang a kolektiv [33] byl navržen algoritmus pro prostorovou detekci anomálií pomocí síťové tomografie, kterou nazvali „anomografie“ spojením slov tomografie a anomálie. Nabízí myšlenku využití síťové tomografie k detekci a odvození anomálií. Stejně jako PCA používá prostorová detekční schémata a jako statistická analýza používá dočasná schémata.

Autoři definují problém odvození síťových anomálií z nepřímých měření linek SD (*Source to Destination*), jelikož anomálie nemohou být v mnoha případech mě-

řeny přímými metodami. Navrhli algoritmus, který sleduje směrování a provoz v síti. Tento algoritmus je schopný zacházet se změnami ve směrování a chybějícími daty v měření, viz popsaná problematika síťové tomografie v kapitole 1.2. Pro samotnou evaluaci výsledků použili síť Abilene a síť ISP Tier-1.

1.3.5 Výčet dalších metod a technik

ASTUTE (*A Short-Timescale Uncorrelated-Traffic Equilibrium*) – byl prezentován autory Fernando Silveira, Christophe Diot, Nina Taft a Ramesh Govindan [34] jako metoda schopná detekovat různé typy síťových anomálií v mnoha malých datových tocích, oproti použití algoritmu Kalmanova filtru v málo velkých datových tocích, který provádí bodový odhad stavů na základě zašuměných výstupů z měření.
IPS Stratosphere – [35], je projektem ČVUT v Praze, autorů Sebastian Garcìa a kolektiv. Autoři využívají Markovovské řetězce pro behaviorální detekci anomálií.

Dodatek o modelech sítí

Existují různé modely sítí a každý tento model má své vlastní výhody a nevýhody. Volbu modelu sítě v rámci výzkumu ovlivňují vybrané parametry provozu. Modely, které nejsou schopny popsat aktuální kvalitu statistických charakteristik mohou výrazně ovlivnit celý výsledek bádání. Neexistuje model, který by mohl být účinně použit pro všechny typy sítí a danou problematiku. Autor Balakrishnan Chandrasekaran [36] v jeho článku „Survey of Network Traffic Models“ uvádí, že například použití Poissona modelu a Markovova modelu ve vysokorychlostních sítích není vhodné. Do modelů použitých k detekci anomálií se dále řadí ON-OFF modely či IPP (*Interrupted Poisson Process*) modely. Další výčet modelů je uveden v tabulce 1.3.

V závěru výzkumu bychom se měli opřít o fakt výsledných testů a porovnání výsledků s nasazením řešení v reálném prostředí.

Tab. 1.3: Přehled modelů

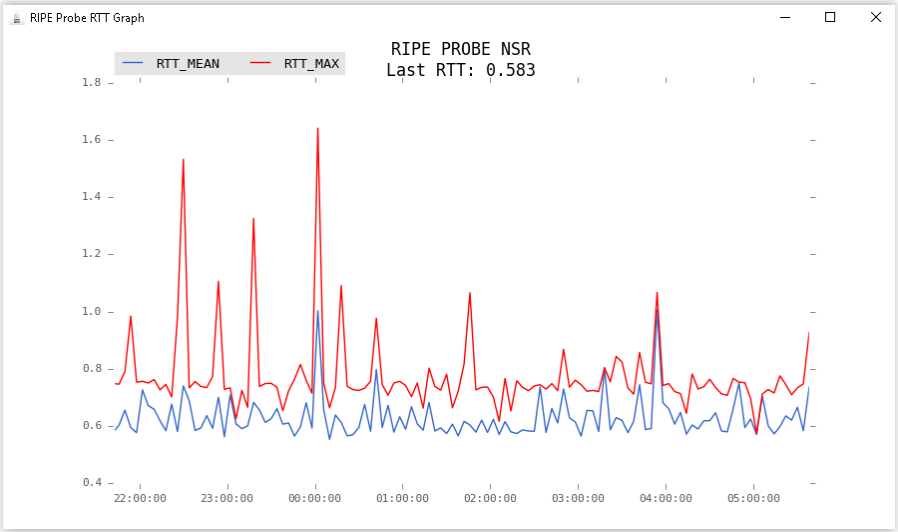
Generativní modely	**Diskriminační modely**
Gaussův model	Logistická regrese
Skrytý Markovův model	Lineární regrese
Naivní Bayesovské	Neuronové sítě
Boltzmannův stroj	Náhodný les
Generativní oponentské sítě [82]	A jiné

1.4 Získávání informací o síti pro účely analýzy

Pro výše popsané techniky je důležitý použitý zdroj dat pro samotnou analýzu. Jako zdroj je možné použít informace o síti a klientech získané skenováním sítě. Skenování sítí je možné rozdělit do dvou hlavních kategorií na pasivní sledování sítě a aktivní skenovaní sítě. Do skupiny pasivního sledování patří zařízení, která jsou vsazena do sítě a pasivně naslouchají na otevřených portech, či poskytují službu, která je určena pro účely detekce.

Aktivní skenování sítě zahrnuje mnoho technik a programů. Mezi základní programy neodmyslitelně patří program PING (*Packet InterNet Gropér*), který byl napsán v roce 1983 Michaelem Johnem Muussem. Pro měření a analýzu sítě používá protokol ICMP (*Internet Control Message Protocol*) a zprávu ICMP echo. V RFC (*Request for Comments*) 1122 [L2] je popsáno, že jakýkoliv host sítě musí zpracovat ICMP echo a odpovědět nazpět zprávou ICMP reply. Samotný protokol je definován RFC 792 [L3]. Na obrázku 1.3 je uveden příklad využití programu PING pro měření obousměrného zpoždění RTT, a to sdružením RIPE, která poskytuje pro účely měření vlastní síťové sondy. Takováto sonda byla v rámci dizertační práce umístěna v laboratoři transportních sítí pro vlastní analytické účely.

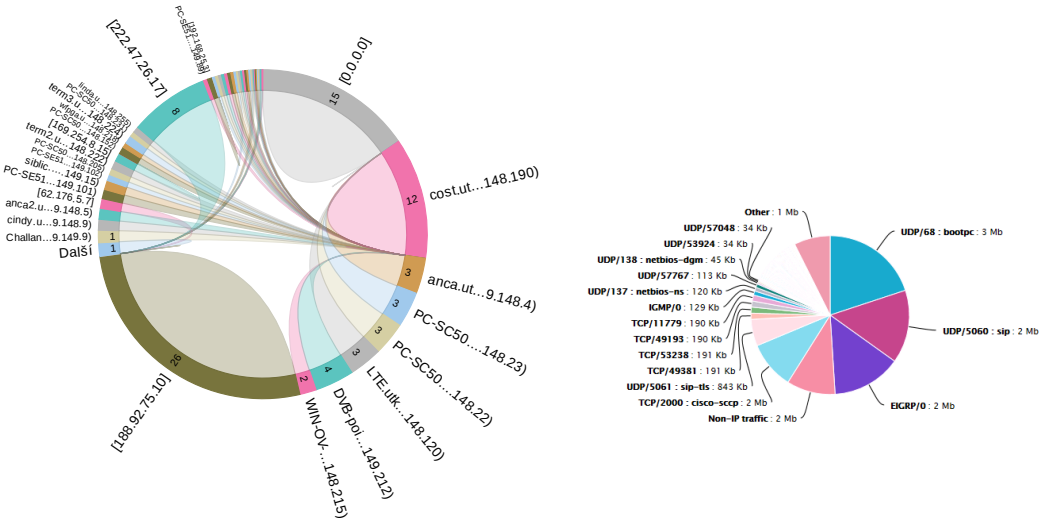
Jako zdroj dat využitelných pro další analýzu slouží také datová úložiště a databázové systémy, které jsou využívány k „těžení“ informací o datech. V určitých typech těchto systému jsou již zabudovány aplikace k vizuální a jiné analýze dat. Takovýmto příkladem je například testované zařízení ENDACE NPB7000 laboratoře transportních sítí, která je součástí centra SIX. Zde byl analyzován vnitřní provoz laboratorní sítě a vizuální výsledek testu je vyobrazen na obrázku 1.4b. Tato sonda pracuje také



Obr. 1.3: Kontinuální ping sondou RIPE

v režimu reálného času, avšak úsudek o chování provozu zůstává na obsluze samotné. Sondy nemusejí být vždy jen hardwarové, často jsou využívány i softwarové nástroje.

Monitorovací program PRTG [37] poskytuje vlastní implementované analytické softwarové sondy, které mohou být nainstalovány na serverech a mohou být vzájemně propojeny. Výstup takovéto sondy je na obrázku 1.4a, který byl pořízen sledováním provozu laboratorního směrovače určeného pro Cisco výuku. Zde by již mohl provoz z IP adresy 188.92.75.10 představovat anomálii. Hloubkovou analýzou bylo zjištěno, že se jedná o skenování portů.



(a) Analýza dat sondou PRTG

(b) Analýza dat sondou ENDACE NPB7000

Obr. 1.4: Grafické výstupy ze síťových sond

Pro analýzu anomálií dat v reálném čase je výhodné použít kombinaci známých postupů a systémů k tomu určených. Tyto systémy se nazývají obecně „hybridní“ systémy. V tabulce 1.4 je uveden stručný výčet nejznámějších výrobců a jejich systémů k detekci a analýze dat. Takovým příkladem je zařízení F5 Networks. Kupříkladu Google Inc. používá vlastní dedikované ASIC pro detekci spamu s přesností 99% v kombinaci s lineární klasifikací, systémů s pravidly (*Rule Based Systems*) a hloubkovým učením (*Deep Learning*) [38, 39].

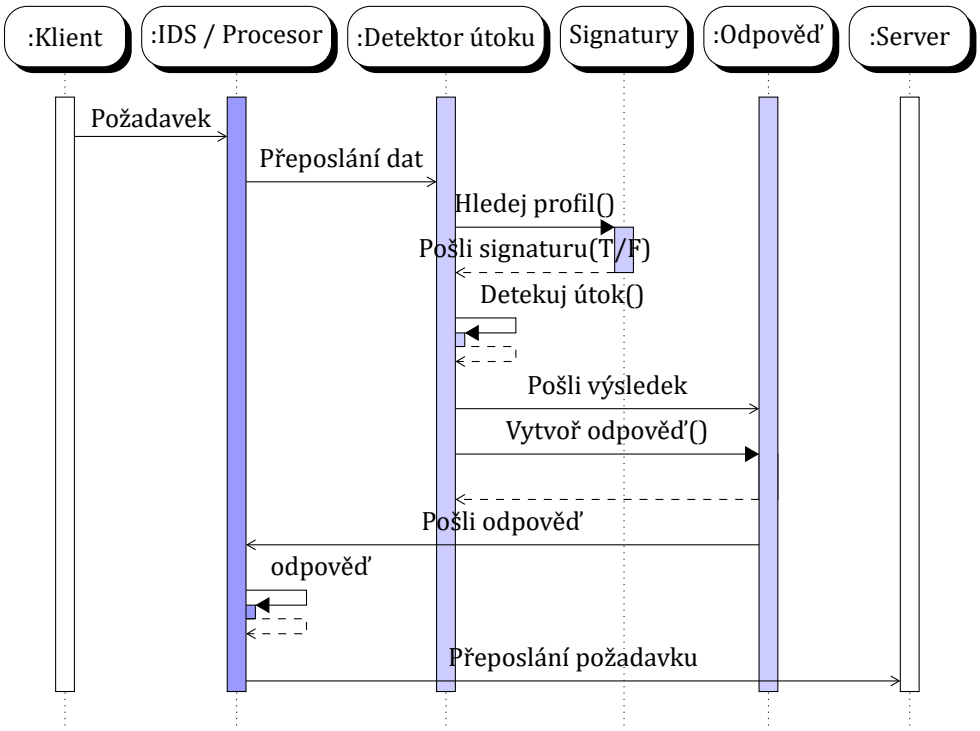
Testován byl také zapůjčený IDS systém GAiA společnosti Check Point Software Technologies Ltd. Jedná se o virtualizovaný systém založený na platformě Cent OS Linux. Výhoda systému je především ve škálovatelnosti a její obsluze. Jednotlivé rozšiřující moduly (*Blades*) jsou zaváděny za běhu celého systému.

Zařízení určená k detekci anomálií síťového provozu jsou označována souhrnným názvem IDS (*Intrusion Detection System*). Podle předchozího popisu mohou být tyto systémy rozlišeny do jednotlivých skupin podle používaných technik. A to do

Tab. 1.4: Příklady systémů detekce a analýzy dat

Název	**Výrobce**	**Používaná technika**
Cisco IPS 4200	Cisco Systems, Inc.	Heuristická analýza
FlowMon ADS	FLOWMON NETWORKS A.S.	Statistické metody
61000 system	Check Point Ltd.	GAiA, kombinace metod
ENDACE NPB7000	Emulex Corporation	Sběr a statistika
Softblade IDS	Barbedwire Technologies	Protokolová analýza
AirDefense Guard	AirDefense, Inc.	Multi-dimenzionální detekce
IBM Threat Protection System	IBM Corporation	Vícecestná detekce
IBM Proventia Network	IBM Corporation	Detekce anomálií
WebSafe, MobileSafe	F5 Networks	DDoS, 7 vrstva
Dedikované ASICs	Google Inc.	Hloubkové učení

skupin abstraktních, signaturních a chování. Příklad sekvenčního diagramu IDS systému používajícího algoritmus založený na signaturách je zobrazen na obrázku 1.5. Zde je příchozí provoz porovnáván s databází jednotlivých signatur. Modul „detekce anomálií“ je možné rozšířit o profily chování, čímž je dosaženo hybridního systému.



Obr. 1.5: Základní sekvenční schéma IDS

Na uvedené systémy jsou kladeny vysoké požadavky jak na přesnost zpracování dat a odpovědí, tak také na rychlost odezvy jednotlivých událostí. Cílem výrobců IDS systémů je maximální možná míra přenesení takového zpracování dat do hardwarové části vyvíjených platforem v kombinaci se softwarovou definicí jejich částí kvůli maximální možné výkonnosti.

Perspektivní se k nasazení v oblasti konvergovaných řešení monitoringu jeví právě SDN technologie, jak ji vnímá například společnost Cisco vývojem XNC technologie a využitím OpenFlow. Firma Radware představila v roce 2013 řešení nazvané „DefenseFlow“ založené právě na řešení XNC. Společným faktorem je škálovatelnost.

Programovatelné hardwarové prvky

Programovatelné prvky sítě hrají významnou roli při tvorbě softwarově definovaných sítí, zabezpečení sítí, monitoringu a aplikací, které jsou v mnoha případech nasazovány dle potřeb zpracování aktuálního provozu. Jednotlivá řešení jsou založena na vývojových platformách. Jedním z předních výrobců takovýchto řešení je firma XILINX [40], která se zabývá jejich vývojem a prodejem.

V oblasti vývoje softwarově definovaných sítí se podílí také společnost Cesnet a spolupracující vysoké školy a instituce. Je zde možno uvést projekt Liberouter [41] a vývoj softwarově definovaného monitoringu, založeného na uvedených FPGA kartách. Tyto programovatelné karty nacházejí například uplatnění v projektech společnosti Cesnet nebo InveaTech [42, 43] v současnosti Flowmon Networks a Netcope Technologies. Slouží zde pro účely monitoringu síťových služeb, směrování, přepínání a využití na poli SDN.

Zmíněná programovatelná hradlová pole FPGA jsou speciální integrované číslicové obvody, které obsahují jednotlivé programovatelné bloky propojené vzájemně maticí spojů. Tyto matice spojů lze konfigurovat jako jeden celek.

Obvody FPGA jsou stále více používány v mnoha dnešních síťových prvcích a jednotlivých inteligentních řešeních vzhledem k jejich klesající ceně, kdy výhody převyšují použití integrovaných obvodů ASIC. Autoři Markos Papadonikolakis a kol. v jejich díle [44] porovnávají efektivitu nasazení paralelních algoritmů v FPGA. Autoři díla dochází k závěru, že vývoj evolučních algoritmů s použitím programovatelných hradlových polí je vysoce efektivní.

TAP, SPAN

Základní všeobecně uznávanou technikou určenou pro sběr dat je využití zařízení TAP (*Test Access Point*) a využití metody zrcadlení komunikace portů síťových zařízení. Například se jedná o funkce SPAN (*Switched Port Analyzer*), RSPAN (*Remote SPAN*) nebo ERSPAN (*Encapsulated RSPAN*) [13] vyvinutých firmou Cisco. Používají

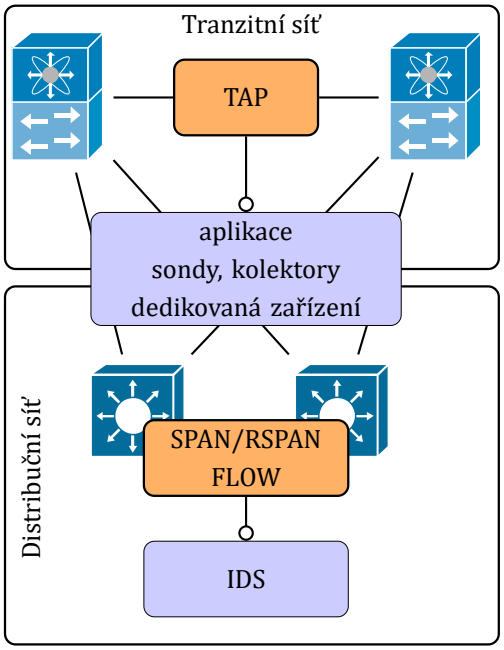
se i jiné technologie, jako jsou například optické senzory nebo protokol pro správu sítí SNMP (*Simple Network Management Protocol*). Na obrázku 1.6 je zobrazeno možné zapojení výše uvedených technologií. Použití RSPAN nebo SPAN funkce přináší i své nevýhody, konkrétně se jedná o zatížení procesorů jednotlivých zařízení, kde je SPAN technologie používána.

Oproti tomu zařízení TAP poskytuje přímé nasazení do spojení, bez nutnosti využití mezilehlých zařízení. Data mohou být přímo přenášena do dedikovaných uzlů, využívajících například IDS systémy. Jedná se o nedisturbidní řešení. Takovéto body jsou pro jiná zařízení „neviditelné“. Což na druhou stranu přináší problémy se škálovatelností celého řešení.

SNMP

SNMP je protokol určený pro monitorování a správu sítí. Umožňuje jak sbírat informace o jednotlivých zařízeních, tak také provádět změny v samotné konfiguraci síťových zařízení. V současnosti jsou definovány celkem čtyři verze. SNMPv1, SNMPv2, SNMPv2c a SNMPv3. RFC k jednotlivým SNMP verzím jsou uvedeny v literatuře SNMP Research International, Inc.[L5].

Nasazení SNMP protokolu rozšiřuje samotnou škálovatelnost, ale opět nelze objektivně sledovat veškeré typy provozu. Striktně centralizované nasazení těchto systémů nemusí přinášet jednoznačnou odpověď na některé typy bezpečnostních incidentů a jiných anomálií provozu.



Obr. 1.6: Základní schéma zapojení technologií pro monitoring

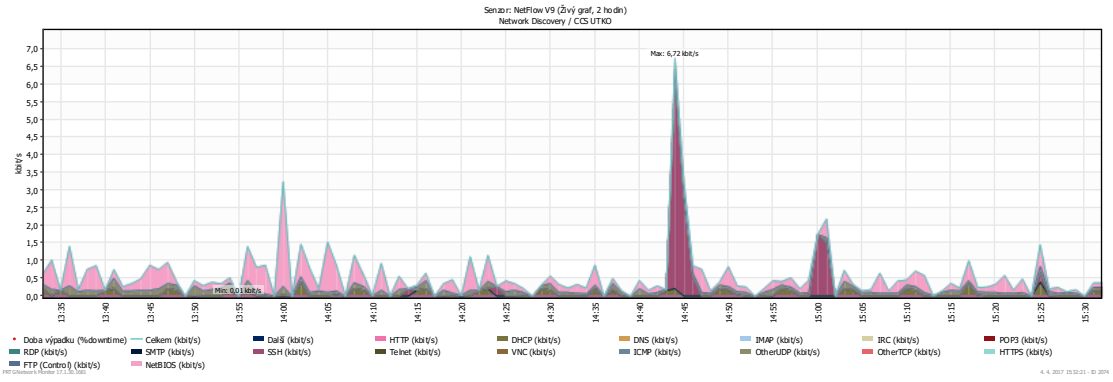
NetFlow

Protokol NetFlow byl vyvinut společností Cisco Systems, Inc. a implementován v jejich síťových produktech [45]. Tento protokol je velmi populární a v různých variantách je používán také mnoha jinými výrobci síťových prvků. Nejnovějším nástupcem je Internet Protocol Flow Information Export (IPFIX). Zpráva protokolu IPFIX se skládá z hlavičky a pole obsahující záznamy o provozu. Tyto zprávy jsou zasílány ze síťových zařízení do analyzátorů (kolektorů).

Příkladem takového kolektoru je Scrutinizer [46], či dříve zmíněný PRTG systém [37] – používá se pro sběr zpráv nejen protokolu NetFlow odeslaných z jednotlivých síťových zařízení. Jsou-li síťová zařízení správně nakonfigurována, jsou pravidelně odesílány informace o tom, které uzly (IP adresy) komunikují. Tyto zprávy obsahují také informace o portech, protokolech a také informace o délce trvání jednotlivých spojení, viz obrázek 1.7.

V případě, že tyto informace o celé síti jsou agregovány na jednom místě, je možné poměrně přesně identifikovat určité typy síťových útoků, typicky například DoS / DDoS, pokusy o záplavu posloupnosti paketů s příznakem SYN (*Synchronize*). Mohou být také identifikované určité typy pokusů o ilegální průnik do sítě a jiné další události. To vše závisí na kvalitě kolektoru a jeho schopnosti statistického zpracování jednotlivých zpráv NetFlow.

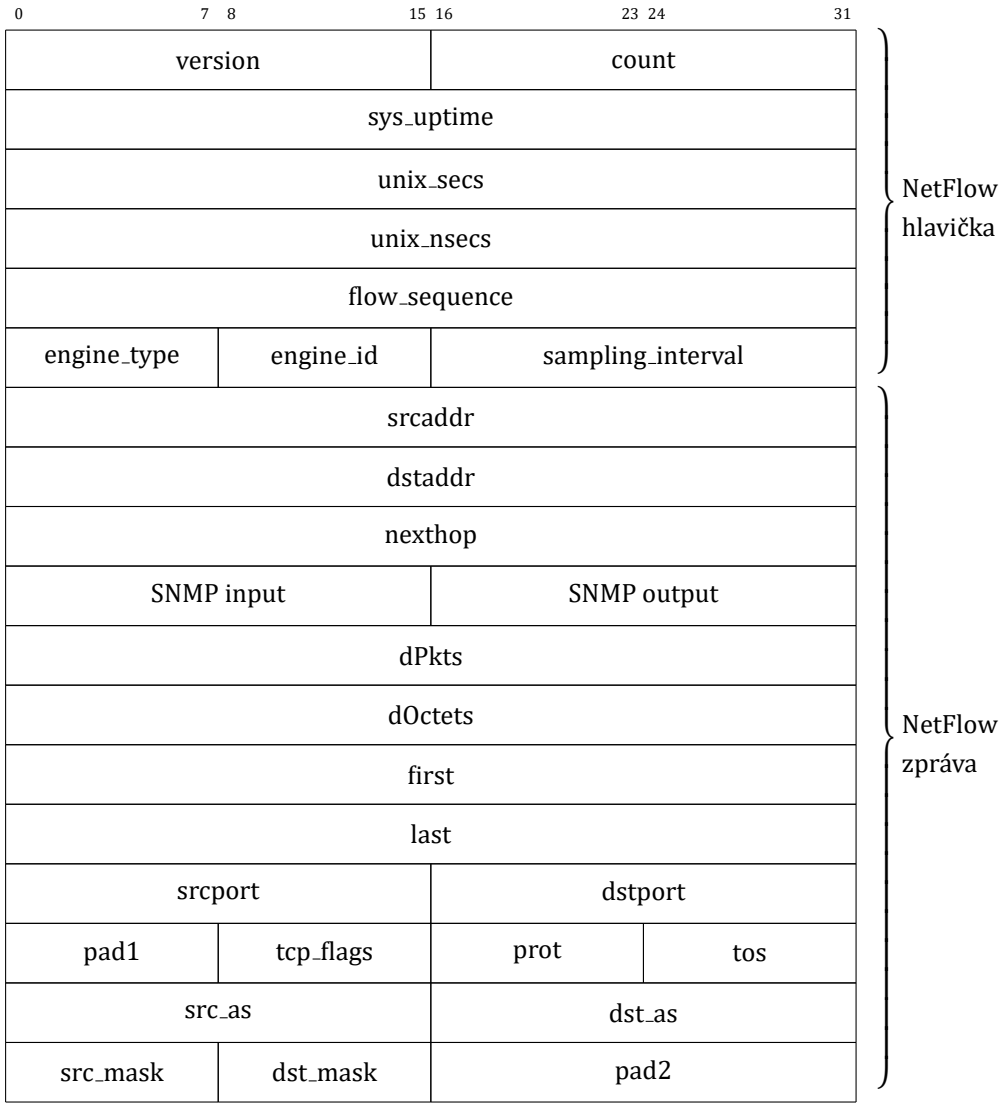
Dalším příkladem použití NetFlow je uchovávání dat „provozních údajů“ s názvem Data Recognition (DR), což je vyžadováno nejen českou legislativou, ale také například evropskou legislativou. V České republice (ČR) se jedná o „zákon o elektronických komunikacích“ (č. 127/2005 Sb), konkrétně § 97 odstavec 3. Tento zákon byl založen původně na směrnici Rady 2006 Evropského parlamentu a Rady/24/ES. Tato směrnice byla zrušena v roce 2013. Evropská legislativa například definuje DR v doporučení R (87) 15, užití osobních dat v policejním sektoru.



Obr. 1.7: Graf PRTG sondy NetFlow verze 9

Je proto pravděpodobné, že značný počet zemí Evropské unie má legislativu, která je podobná legislativě ČR. Podstatou použití NetFlow v DR je uložení NetFlow zpráv a jejich export oprávněným žadatelům v původní – nezměněné podobě. NetFlow je také používán správci velkých podnikových sítí. Používají jej především pro sledování provozu, který uzel (počítač / server) komunikuje v daném čase. Používají jej také jako statistiku, které protokoly jsou nejčastěji zastoupeny a pod. Nejdůležitější součástí kolektorů jsou především algoritmy, které musejí být schopné hlásit pouze skutečné útoky a nevygenerovat velký počet tzv. falešně pozitivních reakcí.

Formát NetFlow verze 5 je uveden na obrázku 1.8, význam jednotlivých položek je pak uveden v příloze v tabulce A.1. NetFlow datagram je tvořen hlavičkou a samotnou zprávou. Jednotlivé verze Netflow se od navzájem odlišují, tak jak byly postupně uzpůsobovány nejnovějším technologiím.



Obr. 1.8: Formát NetFlow export datagramu verze 5 [81]

Verze číslo 1 je dnes málokdy používána. Jedná se o první verzi podporovanou od počátku vývoje Cisco IOS (*Cisco Internetwork Operating System*). Verze 5 přidává informace o autonomním systému a o číslu sekvence provozu. Verze 7 přináší podporu přepínačů Cisco Catalyst. Ve verzi 8 je přidána podpora agregace NetFlow a konečně verze 9 je tvořena rozšiřujícím designem. Figurují zde takzvané vzory, neboli „templates“. Zařízení posílají v datagramu tento vzor, podle kterého sonda ví, jaké informace budou následovat v jednotlivých zprávách.

O IPFIX protokolu bychom mohli říci, že se jedná o verzi číslo 10. Tento protokol je definován v RFC 7011 [L4].

2 TEORETICKÝ ÚVOD EVOLUČNÍCH ALGORITMŮ

Uvedené poznatky pochází zejména z literatury autorů McDonnell a kol. [47] a Hynek, J. [48]. Problematika evolučních algoritmů je velice obsáhlá a evoluční algoritmy jsou stále předmětem výzkumu. Počet publikací v této oblasti roste. Evoluční algoritmy vycházejí zejména z principů evoluční teorie o původu druhů přírodním výběrem, či uchováním prospěšných plemen v boji o život, „On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life“ Charlese Roberta Darwina (rok 1859). Dále také zakladatele moderní genetiky Johanna Gregora Mendela, rodáka z Dolního Slezska, z jeho práce o experimentech na rostlinách „Versuche über Pflanzenhybriden“ (rok 1856).

Motivem k využití daných principů je především jejich aplikace v praktických problémech, které nejsou jinými metodami řešitelné, i přes definici teorému NFL (*No Free Lunch*) v hledání a optimalizaci. Evoluční algoritmy (EA) tedy představují vhodné techniky k řešení složitých optimalizačních úloh, nelineárního programovacího problému.

Prvními pionýry v této oblasti a problematice byli Fraser, Bremermann a Reed v 50tých a 60tých létech dvacátého století. Tabulka v příloze B.1 uvádí stručný přehled těchto algoritmů a principů. Mezi neuvedené evoluční algoritmy dále patří také skupina EDA (*Estimation of Distribution Algorithm*) algoritmů, diferenciální evoluce a neuroevoluce.

Evoluční algoritmy jsou zařazeny do oblasti řešení metaheuristik s populacemi (*Population-based Metaheuristics*). Tento problém vyžaduje najít řešení nestatických proměnných. Pseudokód evolučních algoritmů je uveden níže v programovém kódu 2.1. Je generována pseudonáhodná populace a následně algoritmus prochází jednotlivými stavy EVALUACE, SELEKCE a REPRODUKCE do ukončení vlastní podmínky.

```
generuj náhodně populaci P; t = 0
while True:
        EVALUACE účelové hodnoty (fitness) všech p_i ∈ P(t)
        SELEKCE nejlépe hodnocených p_i ∈ P(t) jako Q(t)
        REPRODUKCE C(t) z Q(t)
    P(t + 1) = C(t)
    t = t + 1
    if cond:
        break
```

Programový kód 2.1: Základní princip EA

Mají za cíl najít takové řešení $\bar{X}$, že provádí optimalizaci funkce, jak uvádí výraz (2.1).

$$\begin{gathered}\text{optimalizace } \bar{X}, \text{kde } \bar{X} = (x_1, x_2, \dots, x_n) \in \mathcal{R}^n \\ \text{a } \bar{X} \in \mathcal{F} \subseteq \mathcal{S}\,,\end{gathered} \tag{2.1}$$

přičemž množina $\mathcal{S} \in R^n$ definuje prohledávaný prostor a množina $\mathcal{F} \subseteq \mathcal{S}$ definuje nejvíce vyhovující prostor z prostoru prohledávaného. Většinou je prohledávaný prostor $\mathcal{S}$ definován jako n dimenzionální prostor v $\mathcal{R}^n$, proměnné jsou definovány jako dolní a horní hranice dle výrazu (2.2), $L =$ levá strana, $P =$ pravá strana.

$$L(i) \leq x_i \leq P(i);\ 1 \leq i \leq n \tag{2.2}$$

Množina $\mathcal{F}$ je definována na prohledávaném prostoru množiny $\mathcal{S}$ a jsou přidány dodatečné omezující podmínky uvedené v (2.3).

$$\begin{aligned} g_j(\bar{X}) &\leq 0; pro\ j = 1, \dots, q \\ h_j(\bar{X}) &= 0; pro\ j = q+1, \dots, m \end{aligned} \tag{2.3}$$

Obecně, evoluční techniky používají k ohodnocení (evaluaci) nejlepšího řešení (jedince) účelovou (*objectives*) funkci f, též nazývanou fitness funkce, (2.4).

$$eval_f(\bar{X}) = f(\bar{X}); pro \bar{X} \in \mathcal{F} \tag{2.4}$$

Dále jsou používány omezující podmínky f_j pro j-tou podmínku pro konstrukci ohodnocení. Tato funkce je definována vztahem (2.5).

$$f_j(\bar{X}) = \begin{cases} \max\{0, g_j(\bar{X}) & \text{pokud } 1 \leq j \leq q \\ |h_j(\bar{X})| & \text{pokud } q+1 \leq j \leq m \end{cases} \tag{2.5}$$

Nejpoužívanější evoluční algoritmy jsou právě algoritmy genetické a kombinované evoluční strategie (ES). Evoluční algoritmy se používají jak pro jednokriteriální optimalizaci, tak také pro vícekriteriální optimalizaci nazývanou také multi-objektivní či více-objektivní optimalizace MOO (*Multi-Objective Optimization*). Inspirovány biologií, evoluční algoritmy převzaly pojmy jedinec, populace a fitness funkce. Jedinec představuje přípustné řešení, skupina jedinců populaci. Ohodnocující funkce určuje kvalitu jedinců a jedná se o optimalizační funkci, kde je hledáno globální maximum nebo minimum.

Diferenciální evoluce

Diferenciální evoluce představuje postup k nalezení minima vícemodálních funkcí pomocí heuristického hledání. Tento algoritmus byl navržen pány Rainer Storn a Kenneth Price [49] v roce 1997, v jejich práci „Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces“. Algoritmus dife-

renciální evoluce je stále velmi populární a je dále rozvíjen. Výsledky experimentů dokazují, že diferenciální evoluce dosahuje lepších výsledků konvergence než jiné stochastické algoritmy.

Nová populace v diferenciální evoluci je vytvářena tak, že postupně pro každého jedince vytvoří konkurenta a do nové populace je vložen jedinec s nižší funkční hodnotou. V práci autorů je pseudokód napsán v jazyce C. Pomoci jazyka Python bychom jej mohli zjednodušeně zapsat následovně:

```
generuj populaci P
while True:
    for i in N:
        generuj vektor u
        vytvoř vektor y křížením u a x[i]
        if f(y) < f(x[i]):
            do Q zařaď y
        else do Q zařaď x[i]
        P = Q
    if cond:
       break
```

Programový kód 2.2: Algoritmus DE

Vícekriteriální evoluční algoritmy

Vícekriteriální optimalizace (MOO) pracuje s více ohodnocujícími funkcemi (*Objectives*) a je schopna nalézt optimální řešení v závislosti všech kritérií, a to simultánně. Tato problematika je detailně zkoumána autory Riccardo Poli a kol. [50]. V tomto díle jsou uvedeny mnohé příklady použití MOO.

Takovýto optimalizační algoritmus pracuje obecně na principu hledání řešení v závislosti na množině rozhodujících proměnných. Tyto proměnné představují omezující podmínky algoritmu a vektory jednotlivých funkcí jsou optimalizovány současně. Takový vektor funkcí obsahující jednotlivé elementy představuje objektivní funkci všech rozhodujících faktorů a vede tedy na neunikátní řešení daného výpočtu. Výsledek takového řešení není v každém případě zcela přesný, ale podává dostatečnou míru přesnosti výsledku.

Do oblasti vícekriteriální optimalizace se řadí algoritmus NSGA-II [51]. Tento algoritmus minimalizuje multidimenzionální funkci aproximací Pareto. Provádí vzorkování prohledávané oblasti a každá tato prohledávaná oblast je nazývána populace. Počet populací je určen parametrem velikosti populace, která je získána vytvořením potomstva z nejlépe vyhovujících kritérií hledaných bodů předchozí populace.

Nejlepší jedinci jsou vypočítáni na základě nedominujících vazeb funkcí vytěsňování vzdálenosti. Cílem je nalézt množinu Pareto optimálních řešení. Základní

matematický vztah optimalizačního procesu hledání maximální hodnoty dané funkce $f(x)$ v závislosti na hodnotě funkce $g(x)$, je definován dle vztahu (2.6).

$$\max\{f(x)|g(x) \geq 0; x \in \bar{X}\}, \tag{2.6}$$

kde $f(\bar{X}) = (f_1(x), f_2(x), \ldots f_i(x))^T$ jsou vektory objektivních funkcí a $g(\bar{X}) = (g_1(x), g_2(x), \ldots g_i(x))^T$ jsou vektory omezujících podmínek daného řešení. Hodnota x představuje vektor vlastního řešení.

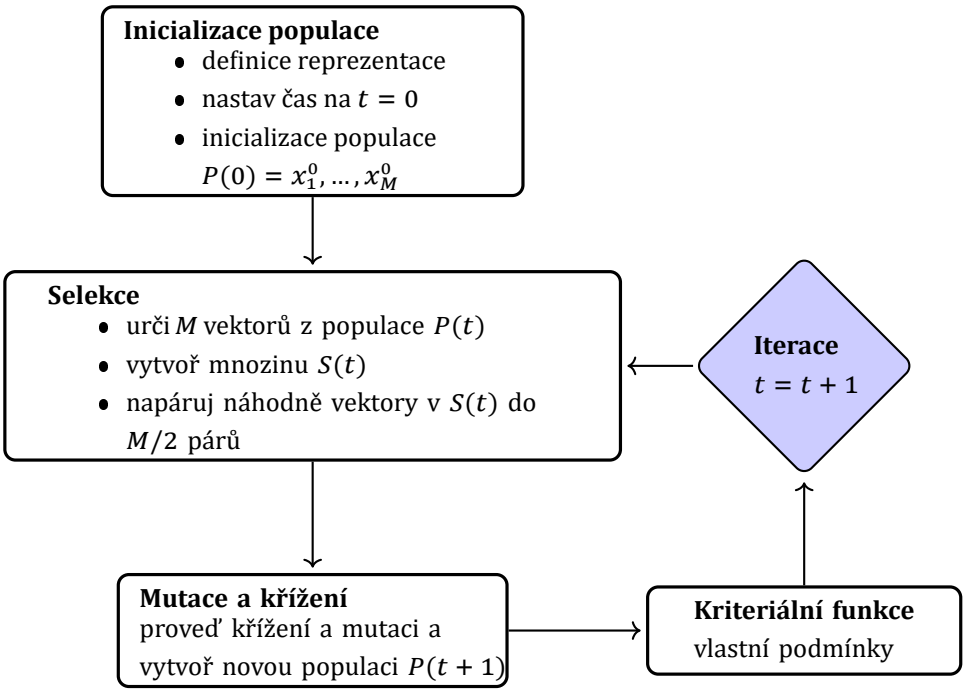
K porovnání algoritmů vícekriteriální optimalizace se používá aproximace Pareto optimální množiny. K tomuto účelu slouží nejčastěji používané indikátory ohodnocení – inverzní generační vzdálenost IGD (*Inverse Generational Distance*) a hyper objem HV (*Hypervolume*).

2.1 Genetický algoritmus

Genetické algoritmy (GA) slouží k nalezení přesného nebo přibližného řešení optimalizační úlohy. Jedná se o metaheuristické metody inspirované evoluční teorií Ch. R. Darwina. Ze stochastických metod využívají prohledávání stavového prostoru, provádí exploraci a z deterministických metod využívají funkce prohledávání nejvíce vyhovujících prostorů tzv. exploataci. Tyto dvě metody tvoří nastavení parametrů GA. V tabulce 2.1 je uveden výčet používaných pojmů GA.

Tab. 2.1: Genetický algoritmus – pojmy

Označení	**Význam**
Jedinec h (*individual*)	Držitel genetické informace
Gen (*gene*)	Pozice informace v chromozómu
Genotyp (*genotype*)	Genetická informace
Fenotyp (*phenotype*)	Konkrétní vyjádření genetické informace
Alela (*allele*)	Konkrétní symbol v chromozómu
Chromozóm (*chromosome*)	Genetická informace jedince
Potomek (*offspring*)	Výsledek rekombinace dvou či více rodičů
Rodič (*parent*)	Jedinec vstupující do rekombinace
Populace p (*population*)	Velikost populace, množina jedinců
Hodnotící funkce f (*fitness*)	$f(h)$ udává kvalitu jedince, míru přizpůsobení
Křížení (*crossover*)	Rekombinace rodičů
Mutace (*mutation*)	Náhodná změna genetické informace
Selekce (*selection*)	Volba jedinců pro další reprodukci
Migrace (*migration*)	Přechod jedinců mezi populacemi
Doplnění (*reinsertion*)	Přesun jedince do nové populace bez křížení
Schéma (*scheme*)	Vzor, či šablona chromozómu



Obr. 2.1: Základní diagram genetického algoritmu

Na obrázku 2.1 je uveden princip GA, kde každý blok provádí jednoduchou operaci. Modul inicializace populace generuje náhodně jedince, vytváří počáteční populaci a předává ji k výběru modulu selekce, modul fitness vyhodnocuje jedince (fenotyp), který je reprezentován genotypem (chromozom). Následně probíhá mutace a křížení jedinců v populaci. Tyto moduly představují tzv. genetické operátory. Celý proces je vyobrazen včetně základního pseudokódu matematického zápisu. Techniky používané pro stanovení omezujících podmínek algoritmu hrají důležitou roli pro jeho celý průběh a mají výrazný vliv na výsledek samotný. Je důležité se vždy rozhodnout, zda zvolit metody k prohledávání prostoru množiny $\mathcal{S}$ nebo zvolit přímé či nepřímé prohledávání množiny prostoru $\mathcal{F}$.

Postup genetického algoritmu je ilustrován na následujícím příkladě. V prvním kroku je vygenerována počáteční populace s velikostí $N = 6$. Každý jedinec je následně ohodnocen. Pokud jedinec disponuje požadovanou vlastností, je jeho genu přiřazena hodnota 1, v opačném případě 0. Celý řetězec genů jedince tvoří chromozom, v tomto případě reprezentován sledem jedniček a nul, tedy v binární reprezentaci viz tabulka 2.2.

Po vhodném ohodnocení jedinců následuje selekce, kde jsou vytvořeny tzv. rodičovské páry. Takovýto výběr by měl být co nejvíce podobný Darwinově teorii a napodobovat co nejpřesněji přírodní principy o původu druhů.

Tab. 2.2: Počáteční populace a její ohodnocení

Jedinec	Chromozóm	Ohodnocení
1	0 0 1 0 0 1	2
2	1 1 0 1 1 1	5
3	0 1 0 0 0 0	1
4	1 0 0 1 1 1	4

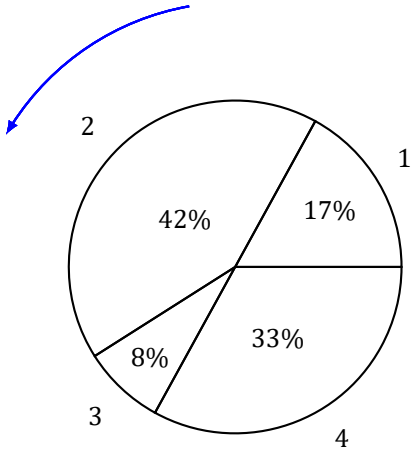
2.1.1 Selekce

Mechanizmus ruletového kola (*Roulette Wheel Selection*) – jedná se o nejrozšířenější formu selekce používající mechanizmu výběru jedince na základě jeho přímé úměry (*Fitness-proportionate Selection*). Jednotlivým jedincům s vyšším ohodnocením je přiřazena větší výseč ruletového kola, je tedy větší pravděpodobnost, že takovýto jedinec bude vybrán. Vhozenou kuličku v tomto případě představuje náhodně generované číslo z rozsahu ohodnocení jedinců. Plocha výseče je přímo úměrná jejímu ohodnocen. V takovémto případě je sečteno ohodnocení všech jedinců a proporcionálně přiřazena výseč. V tabulce 2.3 je uveden celý výpočet.

Z uvedené tabulky vyplývá, že jedinci s číslem 2 bude přiřazena proporcionálně největší výseč kruhové rulety. Grafické zobrazení rozdělení výsečí je na obrázku 2.2. Šipka ukazuje směr vhozené kuličky a sestavení posloupnosti jednotlivých výsečí.

Tab. 2.3: Výpočet ruletového kola

Jedinec	Chromozóm	Ohodnocení	% z celkového ohodnocení	Kumulované hodnoty výseče
1	0 0 1 0 0 1	2	17	0,166
2	1 1 0 1 1 1	5	42	0,582
3	0 1 0 0 0 0	1	8	0,665
4	1 0 0 1 1 1	4	33	1
Celkové ohodnocení		**12**		



Obr. 2.2: Ruletová selekce

Pokud máme takto zkonstruované ruletové kolo, stačí následně vygenerovat reálné náhodné číslo $r \in \langle 0, 1\rangle$ a vybrat i–tého jedince z populace o velikosti N pokud platí vztah (2.7).

$$\sum_{j=1}^{i=1} p_j < r < \sum_{j=1}^{i} p_j, i \in \{1, \ldots, N\} \tag{2.7}$$

Pro zvýšení efektivnosti může být takovéto relativní ohodnocení počítáno z kumulativního ohodnocení, které je definováno vztahem (2.8), převzatým z [48].

$$\bar{f}_i = \sum_{j=1}^{i} p_j = \sum_{j=1}^{i} \frac{f_j}{\sum_{k=1}^{N} f_k}, i \in \{1, \ldots, N\} \tag{2.8}$$

Potom je i–tý jedinec z populace o celkové velikosti N vybrán při platnosti vztahu $\overline{f_i - 1} < r < \bar{f}_i, i \in \{1, \ldots, N\}$. Pro zvýšení pravděpodobnosti přežití nejlepších jedinců může být uplatněn princip přímé selekce prvního páru rodičů a jejich zkopírování do nové populace, tzv. elitismus. Zde však existuje riziko uvíznutí algoritmu v lokálním maximu.

Další typy selekcí

Mezi další typy selekce patří stochastické univerzální vzorkování (*Stochastic Universal Sampling*). Jedná se o variaci na ruletovou selekci, patří také do skupiny výběru jedince na základě jeho přímé úměry. Mezi další typy se řadí turnajová selekce (*Tournament Selection*), selekce pomocí odměn (*Reward–based Selection*) a mnohé další,

jako jsou selekce pomocí proporcionálního ohodnocení, selekce pomocí exponenciálního ohodnocení, či selekce zkrácením.

Selekce zkrácení (*Truncation Selection*) je nejjednodušším typem selekce, která představuje pravděpodobně nejméně používanou strategii výběru. Selekce zkrácení jednoduše zachovává určité nejlepší procento populace. Například, je možné zvolit nejvhodnějších 25 % z populace 100 jedinců. V tomto případě se vytvoří čtyři kopie každého z 25 jedinců tak, aby byla zachována populace 100 jedinců.

Selekce podle pořadí (*Rank Selection*) je podobná mechanizmu výběru jedince na základě jeho přímé úměry s tím rozdílem, že pravděpodobnostní výběr je úměrný relativní vhodnosti než absolutnímu ohodnocení.

Mechanizmus selekce podle pořadí slouží k tomu, aby se zabránilo předčasnému selekčnímu tlaku pro velké rozdíly v ohodnocení jedinců, které se vyskytují v prvních generacích. Zesiluje malé rozdíly v ohodnocení jedinců v pozdějších generacích. Selekční tlak se zvyšuje ve srovnání s alternativními výběrovými strategiemi.

Sigma Scaling stejně jako selekce podle pořadí vyvíjí středně selekční tlak v průběhu času tak, že není příliš silný v prvních generacích, ale také ne příliš slabý poté, co se populace stabilizuje a rozdíly ohodnocení jsou menší. Řecké písmeno Sigma se používá ve statistice, které naznačuje směrodatnou odchylku. Tento stejný význam má i zde. Směrodatná odchylka populace se používá pro kalibraci, takže selekční tlak je relativně konstantní po celou dobu životnosti evolučního programu.

2.1.2 Křížení

Křížení představuje další z kroků genetického algoritmu nutného pro vývoj populace. V případě dvou jedinců v populaci reprezentovaných binárním zápisem BIN_L a BIN_P; $0 \leq BIN_L, BIN_P \leq \sum_{n=1}^{N} D_n$, kde D_n je délka chromozomu, potom operátor křížení rozdělí populaci chromozomů jedinců do párů a vymění prvních BIN_L bitů chromozomu jedince s posledními bity BIN_P chromozomu jedince s definovanou pravděpodobností p_{BIN}, případně o pevně stanovené délce. Záleží na použitém typu křížení. Následující tabulka shrnuje typy používaných křížení (rekombinací).

Aritmetické křížení[1], které bude použito dále v práci, je tvořeno náhodným zvolením dvou rodičů $R1$ a $R2$ a vygenerováním náhodného čísla $\lambda = rand(0,1)$. Následně je provedeno křížení a vytvoření potomků $P1$ a $P2$ dle (2.9).

$$\begin{aligned} P1 &= (\lambda)R1 + (1-\lambda)R2\,, \\ P2 &= (1-\lambda)R1 + (\lambda)R2 \end{aligned} \tag{2.9}$$

[1] V některé literatuře označované jako konvexní kombinace.

Tab. 2.4: Přehled typů křížení v GA v závislosti na formě jedince

Všechny formy jedince	Reálné hodnoty	Binární reprezentace	Jiné typy křížení
Diskrétní křížení	Střední rekombinace	Jednobodové nebo vícebodové křížení	Aritmetické křížení
	Řádková rekombinace	Uniformní křížení	Heuristické křížení
	Rozšířená řádková rekombinace	Křížení mícháním	

2.1.3 Mutace

Pro každou generaci $t \in \{1, 2, \ldots, T\}$ je po provedeném křížení provedena mutace s předdefinovanou pravděpodobností. V případě binární reprezentace jedince je pro každou binární hodnotu v chromozomu jedince provedeno přepsání hodnoty 0 na hodnotu 1 a opačně, dle zvolené strategie. Mutace je jedním z operátorů a je potřebná pro zajištění diversity. Může přinést zlepšení výsledku, ale i naopak nemusí. Závisí na řešeném případě.

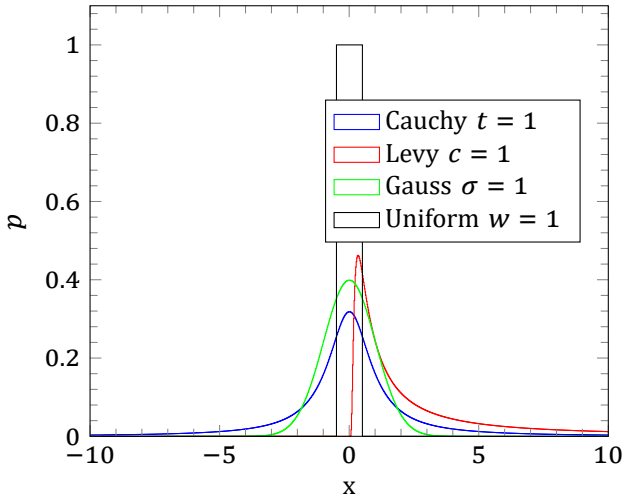
Opět zde existuje několik způsobů, či typů mutací – jak rozprostřít změnu mezi jednotlivé geny chromozomu. Mezi tyto patří binární mutace, mutace reálných čísel, flip-bit mutace, neuniformní, uniformní, Gaussova. Nejlepších výsledků dle prozkoumané literatury [52] dosahuje autoadaptivní Gaussova mutace, která používá normované rozprostření změny v chromozomu. Obecně princip mutace lze ukázat na následujícím programovém kódu 2.3.

```
def mutace(populace):
    nova_populace = np.multiply(
                    populace, np.random.randint(2, size=(velikost_pop, delka_chromozomu))
            if random.random() < hodnota
            else np.ones((velikost_pop, delka_chromozomu), dtype=np.int))
    return nova_populace
```

Programový kód 2.3: Mutace populace

V prvním kroku vstupuje do funkce celá populace. Nová populace je vytvořena multiplikací pomocí jedničkové matice, nebo jedničkové a nulové matice, podle toho, zda náhodně generované číslo je menší nežli pevně zvolená hodnota. V tomto případě se jedná o mutaci binární.

Rozprostření podle Gaussova by bylo možné například pro reálná čísla aplikovat pomocí funkce `random.gauss(mu, sigma)`. Na obrázku 2.3 jsou zobrazeny jednotlivé hustoty pravděpodobnosti pro jednotlivá rozdělení.



Obr. 2.3: Porovnání rozdělení hustoty pravděpodobností

Hustota pravděpodobnosti Gaussova (normálního) rozdělení je vyjádřena dle vztahu (2.10).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{2.10}$$

kde v případě uvedeného příkladu se $\sigma = 1$ a $\mu = 0$. V případě použití takového rozdělení budou čísla distribuována vzhledem k této distribuční funkci.

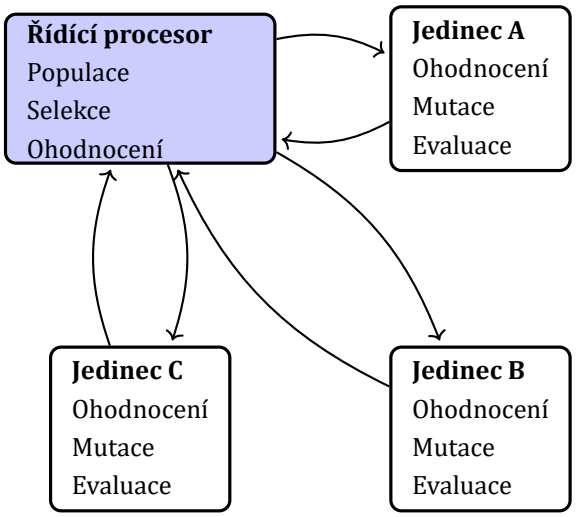
2.2 Typy genetických algoritmů

Existují různé modifikace genetických algoritmů. Obecně se dají rozdělit na globální, migrační a difuzní [53]. Mezi ně se řadí sekvenční, hybridní, adaptivní, integrované a jiné. V následující podkapitole je uvedena teorie z oblasti paralelizace, která zvyšuje výkonnost genetických algoritmů.

2.2.1 Paralelní GA a paralelní implementace

Z důvodů potřeby urychlení zpracování výpočtů genetických algoritmů vznikly GA paralelní. Jsou zde rozlišovány jednotlivé druhy paralelizace jako je například paralelizace ohodnocující funkce, paralelizace populace a ostrovní model (*Island Model*). Na obrázku 2.4 je znázorněn princip paralelizace ohodnocující funkce. Genetický algoritmus je zpracováván na řídícím procesoru a jednotlivé podružné procesory pouze ohodnocují jednotlivé jedince a navrací toto ohodnocení řídícímu procesoru.

Paralelní zpracování genetických algoritmů přináší také mnohé otázky řešení způsobu jejich implementace. Jedním z problémů je otázka nadměrného urychlení zpracování algoritmu (*Speed-up Problem*). Teoreticky je uvažováno, že dolní hranice



Obr. 2.4: Paralelní zpracování GA

rychlosti výpočtu při paralelizaci algoritmu je závislá na počtu použitých procesorů. Zjednodušeně lze říci, že pokud trvá použitému procesoru výpočet po dobu T, pak při použití více stejných procesorů $PROC$ a rozložení výpočtu do paralelní úlohy bude tato doba T trvat $T/PROC$ a více. Autor Enrique Alba [54] se ve svém díle právě věnuje otázce použití paralelních implementací v závislosti na otázce nadměrného urychlení a poskytuje provedená měření.

Otázka paralelizace je důležitým faktorem samotných GA a přináší důkazy měření, že čas zpracování nezávisí pouze na počtu použitých procesorů. Samotná doba výpočtu může být mnohem nižší než se předpokládá při teoretické reprezentaci paralelních GA. Efektivní paralelizace se dá dosáhnout implementací do hardwarové části prvků.

Genetický algoritmus v programovatelných prvcích

První větší zmínkou použití genetických algoritmů v FPGA je publikace autorů Scott a kol. [55]. Při realizaci samotné rozdělili algoritmus do menších modulů naprogramovaných ve VHDL jazyce pomocí programovatelné desky XC4000 BORG. Pro selekci rodičů v GA použili autoři funkci rulety a jednobodovou metodu křížení. Tato implementace byla však omezena na pevnou velikost populace. Autoři zde však poukázali na zásadní možnost implementace GA v FPGA.

Většina publikací uvádí fixní parametry velikosti populace, fixní počet generací GA a poměry křížení. Základním vylepšením je přístup korekce fitness funkce během běhu programu. Autoři Fernando a kol. [56] přináší ve svém díle řešení s využitím Virtex-II Pro FPGA Board. Jejich práce dovoluje implementaci paralelizace GA.

3 STĚŽEJNÍ PUBLIKACE K DANÉ PROBLEMATICE

Mimo publikace (články, knihy a konferenční příspěvky) průběžně citované v dizertační práci, daný výzkum také navázal na následující publikace rozdělené na jednotlivé problematiky.

Síťová tomografie a její principy v datových sítích

V rámci dizertace byly zkoumány jednotlivé směry prací superpočítačového centra kalifornské univerzity CAIDA, úzce spolupracující s komunitou RIPE NNC. Tyto práce jsou veřejně dostupné v uvedeném odkaze [57]. Techniky síťové tomografie jsou zde využity v díle „*Challenges in Inferring Internet Interdomain Congestion*“. Autoři zde navrhují řešení validace zahlcení sítě na základě metod síťové tomografie.

Této části se také týká publikace autorů Zhang, Y., Ge, Z., Greenberg, A. a Roughan, M. „Network anomography“ [33].

Evoluční algoritmy

Existuje mnoho výborných publikací zabývajících se obecně problematikou evolučních algoritmů. Významná část v práci byla čerpána z literatury „Genetické algoritmy a genetické programování“ od autora Josefa Hynka [48]. Dále byla práce inspirována knihou „Genetic Algorithms for Control and Signal Processing“ [53]

Principy detekcí provozu

Oblast detekce provozu, přesněji řečeno detekce anomálií provozu, má rozsáhlé spektrum možností aplikace. V této oblasti bylo zásadních prací hned několik. Jednak se týkaly samotné možnosti nasazení genetických algoritmů v oblasti detekce anomálií provozu a dále obecně možnostmi a principy detekcí anomálií provozu. Práce týkající se tohoto tématu jsou: „An Entropy-Based Network Anomaly Detection Method“ autorů BEREZIŃSKI, Przemysław, Bartosz JASIUL a Marcin SZPYRKA [19], „ASTUTE: Detecting a Different Class of Traffic Anomalies“ autorů Silveiray, F ., Diot, C ., Taft, N ., Govindan, R. [34]. Dále to jsou „Network Traffic Anomaly Detection“ [23], „A Genetic Algorithm for Solving the Euclidean Distance Matrices Completion Problem“ [32].

4 SPECIFIKACE VÝZKUMU

Téma dizertační práce je velice obsahově široké, proto bylo nutné se zaměřit na specifickou oblast daného výzkumu. Z prostudované literatury a problematiky byl postupně ujednáván její rozsah.

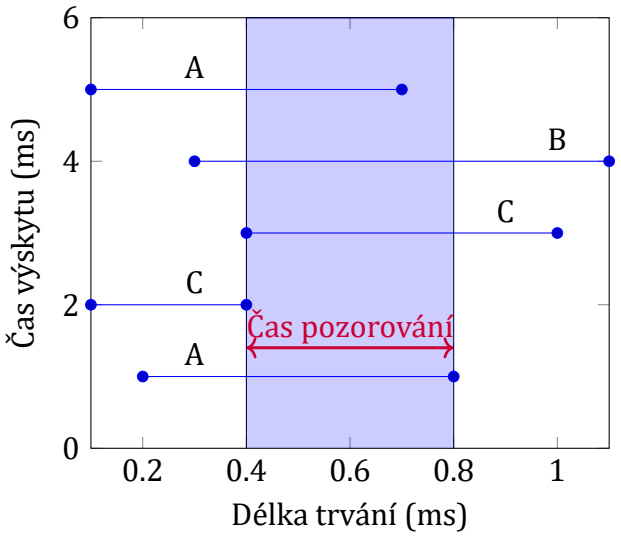
1. Konvergované sítě obecně představují rychle se rozvíjející oblast. Pro vlastní účely práce byl zvolen princip sběru dat pomocí protokolu NetFlow, a to také s ohledem na možnosti a vybavení laboratoře, kde byl výzkum prováděn.
2. Tématika bezpečnosti sítí stále narůstá na významu a konvergované sítě jsou nedílnou součástí této tématiky. Do této tématiky jsou zařazeny problematiky predikce, detekce a obrany. Z pohledu bezpečnosti sítí byla zvolena problematika detekce, která je základním předpokladem pro řešení predikce a obrany.
3. Z oblasti bezpečnostní detekce byla zvolena tématická sekce detekce anomálií. Ač anomálie mohou představovat jakékoliv tendence, odchylky či směry, práce se soustředí na možnosti detekce anomálií jako takových a možnosti nasazení evolučních algoritmů a statistických metod.
4. Z pohledu neprobádaných směrů je práce soustředěna na životnost a životní cyklus anomálního jevu. Pro snadnější identifikaci a prezentaci výsledků bylo zvoleno téma botnetových sítí. Toto téma je ovšem také velice obsáhlé, proto se práce soustředí na řídicí a kontrolní zprávy šířené mezi napadenými stanicemi a řídicími servery.

Ad. 1–3. Analýza přežití

Metoda analýzy přežití byla původně vyvinuta pro měření doby životnosti jedinců. Řadí se také mezi statistické metody, jak o nich bylo pojednáno v úvodní části. Tato analýza může být aplikována na jakýkoliv časový proces, jako je například návštěvnost internetových stránek. Začátek trvání procesu je příchod nového návštěvníka na webové stránky a koncem doby je jeho odchod. Jedním z cílů analýzy přežití je extrakce modelů z dat, která přibližují rozložení doby životnosti. Těmito modely lze odhadnout čas, kdy dojde k události ke vztaženému objektu.

V této metodě je definována levá a pravá cenzura. S pomocí pravě cenzurovaných jedinců jsou známy pouze jejich aktuální hodnoty trvání životního cyklu. Na druhou stranu, při využití levě cenzurovaných jedinců nemáme informace o době jejich vzniku (startu, zformování). Posledním typem je cenzura intervalová. V této cenzuře není znám přesný čas výskytu jedinců a je zjištěn jen částečně.

Analýza přežití je velmi užitečná metoda k porozumění trvání a průběhu procesů a životností. Pro představu, na obrázku 4.1 je příklad s různými kombinacemi



Obr. 4.1: Princip pozorování událostí výskytu

událostí, jejich start, konec a trvání. Každé písmeno A, B, C zde reprezentuje jiný typ komunikace v síti. Čas pozorování představuje cenzuru intervalovou.

Funkce přežití může být nekonečná, ne však negativní. V tomto případě nekonečnost představuje nezapočatou komunikaci. Funkce přežití $S(t)$ je definována dle vztahu (4.1). Tato funkce definuje pravděpodobnost, že v čase t ještě nenastal konec události, nebo ekvivalentně, pravděpodobnost přežití alespoň do doby t.

$$S(t) = Pr(T > t)\,, \tag{4.1}$$

kde platí podmínky $0 \leq S(t) \leq 1$ a $S(t)$ je nezvyšující se funkcí t, protože kumulativní distribuční funkcí T je $F_T = 1 - S(t)$.

Pro odhad funkce přežití je používána metoda parciální věrohodnosti Cox nebo Kaplan-Meier analýza (4.2).

$$\hat{S}(t) = \prod_{j=1}^{k} \left(\frac{n_j - d_j}{n_j} \right) = \hat{S}(t) \left(1 - \frac{d_t}{n_t} \right), \tag{4.2}$$

kde d_j koresponduje s počtem událostí, případně s počtem ukončených událostí v čase j, zatímco n_j je vztaženo k počtu objektů, které jsou stále pozorované v čase j.

Porovnání dvou a více skupin

Test shody (*Compliance Tests*) je použit pro porovnání, zda se dvě či více testovaných skupin shodují. Existuje více takovýchto metod, může být zmíněn například „Breslow“ test nebo „Tarone–Ware“ test. Nejznámějším testem je neparametrický

„Mantel–Cox“ test, pojmenovaný podle jejich autorů N. Mantel a D. Cox. Občas je také nazýván jako „log rank test“. Mantel-Cox Chi-Squared test je definován dle vztahu (4.3), [58]:

$$\chi^2_{MC} = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_2 - E_2)^2}{E_2}, \tag{4.3}$$

kde O_1 je suma výskytů událostí pro experimentální skupinu a O_2 je suma výskytu událostí kontrolované skupiny. E znamená očekávanou sumu pro každou ze skupin. Tento test je také generalizován pro testování více než dvou skupin. Uvedený vztah (4.3) je rozšířen o $k - 1$ definicí na $\chi^2_{MC} = k_1 + k_2 + k_n$.

Ad. 4. Specifikace botnetové sítě

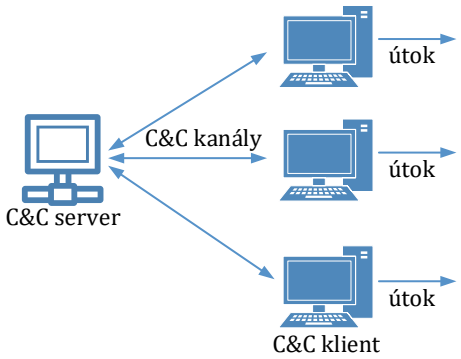
Specifický výzkum je zaměřen na vývoj algoritmu pro detekci anomálií. V současnosti tvoří provoz botnetových sítí bezpečnostní slabinu. Z těchto sítí jsou prováděny útoky či sdílení nebezpečných obsahů. Tento provoz je nutné vnímat jako anomálii, kterou je nezbytné analyzovat a specifikovat. Jednotlivé botnet sítě jsou rozlišovány dle jejich vzhledu a typu použitého protokolu pro komunikaci.

Autor Silvia a kolektiv [59] definoval jednotlivé typy botnet sítí a jejich chování. Jeden z nejstarších typů této sítě využívá pro komunikaci protokol IRC (*Internet Relay Chat*). Mezi další typy se řadí: HTTP (*Hypertext Transfer Protocol*), P2P (*Point to Point*) a HTTP2P provoz v kombinaci s centralizovanou a decentralizovanou správou.

Tento výčet výše uvedených kategorií není jediný. Obecně platí, že botnetové sítě mohou být řízeny specifickým řídícím serverem, který využívá vlastní typ přístupu, neboli získání „root“ práv síťového zařízení. Tento přístup může být zajištěn pomocí SSH (*Secure Shell*) připojení, nebo například pomocí zneužití poštovního serveru. Taková technika je také použita v rámci projektu GCAT [60], který používá účet Gmail k vytvoření C&C (*Command and Control*) kanálů.

Vlasntní C&C kanály zajišťují příjem a odesílání řídicích příkazů a informací mezi řídicím C&C serverem a infikovanými klienty, jak je znázorněno na obrázku 4.2. Řídicí server je schopen řídit mnoho klientů v krátkém časovém období. Na základě příkazů řídicího serveru je možné provést masivní útok typu DDoS.

Vzhledem k výše uvedenému, je rozlišováno mnoho druhů stávajících přístupů a metod jak takový botnet vytvořit. Proces převzetí kontroly nad síťovými zařízeními není pevně definován a může být proveden v zásadě jakýmkoliv individuálním přístupem, a to i pomocí naprogramování vlastního řešení. Takovéto řešení je poté pro ostatní neznámé a nepředvídatelné.



Obr. 4.2: Jednoduchá centralizovaná síť botnet

Algoritmy botnetů

Mnoho botnetů využívá algoritmu generování domén DGA (Domain Generation Algorithm) [61], kdy jsou řídicím serverem generovány a registrovány domény, někdy až tisíce denně. Malware skenuje síť a v případě, že zjistí dostupnost domény generované řídicím serverem se připojí a čeká na instrukce. Obvykle se k vytváření domén používá časová značka. Například ale Torpig botnet využívá názvy témat z uživatelských účtů Twitteru.

Mnoho detekcí je založeno na hashování souborů. Pro každý soubor označený jako malware je vytvořena jedinečná detekce. Algoritmy botnetu na tento princip detekce reagují nepatrnými změnami ve vlastním kódu, aby změnily hodnotu hash pro soubor použitý detektory. Obecně se takovéto botnety nazývají polymorfní. V některých případech nelze vzory či modely provozu použít.

S cílem zabránit mitigačním zařízením převzetí řízení botnetových sítí jsou udržovány algoritmy botnetů partnerské seznamy a ohodnocení spolehlivosti. Některé botnety analyzují prostředí ve kterém se nacházejí a mění chování za účelem vystupovat jako jiný software.

Některé detekce botnetu jsou založeny na metodikách chování nebo popisu C&C infrastruktury. Problém této detekce je v tom, že provoz samotného botnetu je korelován s ostatním provozem a chování tohoto škodlivého provozu může být podobné jako chování provozu „normálního“.

Definice životního cyklu provozu přináší nový aspekt do možnosti detekce botnetu. Může být přínosem v samotné filtraci provozu a následném jejím rozboru.

5 TESTOVÁNÍ PROGRAMŮ PRO VÝVOJ A IMPLEMENTACI ALGORITMU

V rámci disertační práce byly testovány programy, programovací jazyky, které jsou vhodné k vlastní implementaci a modelování navrženého řešení. Byly zjišťovány možnosti využití vývojových programů KNIME, MATLAB, R, SCILAB a PyCharm Professional. Dále také bylo uvažováno využití programů OPNET IT Guru Academic Edition a OMNeT++ [62]. Pro pravděpodobnostní analýzu to byla skupina programů ReliaSoft [63]. Nejvhodnějším pro vlastní implementaci byl zvolen skriptovací jazyk Python a program PyCharm Professional. Z vlastního provedeného testování programu ReliaSoft byla převzata idea implementace části analýzy ve zvoleném jazyce Python.

5.1 Simulační prostředí a pracovní nástroje

Pro vývoj algoritmu a jeho vhodného nasazení bylo provedeno testování programů MATLAB, SCILAB, R, KNIME, OMNeT++ a PyCharm Professional. Toto testování probíhalo se stanovenými kritérii a jejich možnou vzájemnou kombinací:

1. Podpora evolučních algoritmů, zejména pro optimalizační úlohy.
2. Podpora klasifikačních úloh.
3. Podpora síťových prvků s možností vytvoření síťové komunikace a implementace evolučních algoritmů.

První kritérium bylo zvoleno na základě podstaty účelu evolučních algoritmů. Tyto algoritmy jsou určeny převážně pro optimalizační úlohy. Mohou být použity i pro klasifikační úlohy. Ty jsou ale většinou používány například uvnitř tradičních klasifikačních algoritmů, jako jsou neuronové sítě a slouží pro nastavení parametrů těchto algoritmů, případně jsou použity vně neuronových algoritmů pro výběr jejich funkcí. Další podmínkou je také možnost implementace a testování vlastního algoritmu.

Jako první program pro testování byl zvolen KNIME, a to z důvodu předchozích zkušeností s tímto programem. V tomto programu je možné využít již obsažené moduly v jednotlivých repositářích pro vlastní návrh schéma.

Podle vlastních zjištění je zde podpora optimalizačních úloh zastoupena modulem `Multiobjective Subset Selection` v repositáři `Optimization`. Tento modul je vhodný pro úlohu selekce podmnožiny (*Subset Selection*). Optimalizuje určitá kritéria na základě podmnožin zapsaných v řádcích. Konkrétně se jedná o algoritmus NSGAII (*Non-dominated Sorting GA II.*). Představuje konkrétní vytvořený optimalizační prvek. Žádné další rozšiřující prvky, či modely nejsou v programu KNIME pro

definované podmínky vytvořeny. Nejsou zde plně zastoupeny žádné jiné modely klasifikačních úloh, či definované moduly síťových prvků. Ukázalo se tedy, že pro vlastní záměr není tento program dostačující.

V druhé řadě byla testována implementace algoritmů ve vývojovém programu SCILAB, který představuje koncepčně program MATLAB. Je zde zastoupen modul pro převod vytvořených projektů v programu MATLAB do tohoto programu. Je zde možné instalovat přídavné prvky a aplikace pomocí manažera aplikací `ATOMS Module Manager`. Zde byly vybrány a doinstalovány doplňující moduly `Network Topology Generator`, `Network Analysis and Routing eVALuation`. Dále je zde zastoupena celá sada podpory pro optimalizační úlohy, jako jsou genetické a více-objektivní algoritmy. Dále je také možné vytvořit vlastní aplikaci a vlastní doplňující moduly.

Pro možnost testovat vlastnosti sítě a vlastní algoritmy je velmi slibným uvedený modul `Network Analysis and Routing eVALuation`, nazvaný zkráceně názvem NARVAL. Jedná se o projekt univerzity „University of Luxembourg“, výzkumné skupiny SnT (*Interdisciplinary Centre for Security, Reliability and Trust*), uvedený na konferenci SCILABTEC [64] v roce 2014. Tento modul je zaměřen na analýzu síťových protokolů a algoritmů. Dovoluje provádět výzkum směrovacích algoritmů a poskytuje tvorbu síťových grafů, výpočty jednotlivých směrovacích algoritmů a statistickou analýzu.

Python byl vybrán jako jazyk pro implementaci, protože je vhodný pro rychlé vytváření prototypů. Jako Matlab, Python je jazykem skriptovacím. Obsahuje také mnoho knihoven třetích stran, které výrazně pomáhají zkrátit dobu implementace. Jednou z těchto knihoven (balíčků) je DEAP (*Distributed Evolutionary Algorithms in Python*). Využití knihovny DEAP se ukázalo být dostatečně flexibilní pro daný úkol, protože umožňuje uživateli definovat své vlastní algorimty pro výběr, křížení a mutaci, jakož i kombinaci s jinými knihovnami. V samotném řešení práce však byl použit algoritmus vlastní. Bylo využito knihovny Matplotlib pro grafické ztvárnění, generování grafů ze získaných dat.

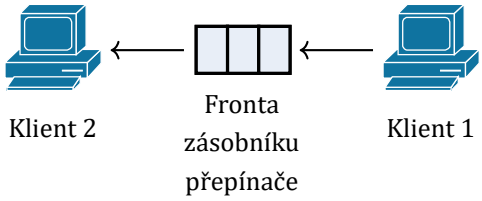
5.1.1 Srovnávací test programu OMNeT++ a Python

Při výběru vhodného nástroje pro návrh algoritmu a jeho implementaci byl proveden srovnávací test dvou vybraných programů OMNeT++ a Python, ve kterých je možné simulovat a testovat provoz síťových zařízení. Konkrétně se jednalo o Python verze 3.4.3, SimPy 3.0.8 a OMNeT++ 4.6 IDE (Integrated Development Environment)

Za tímto účelem byl použit počítač s procesorem Intel(R) Core(TM) i5-3340M CPU @ 2.70GHz, 4 GB RAM a Windows 7 OS $F\times64$. Jako IDE pro Python byl využit PyCharm Educational Edition 1.0.1. Nebyla zde použita žádná knihovna pro zvýšení výkonnosti, jako je například `Parakeet runtime compiler`. Postup provedeného

testu je publikován v [A9]. Účelem bylo zjištění více faktorů, které ovlivňují výběr vhodného nástroje vzhledem k individuálním potřebám a schopnostem.

Základem uvedeného testu je jednoduchý scénář komunikace dvou uzlů sítě (PC) propojených přepínačem, viz obrázek 5.1. Tyto dva uzly navzájem komunikují pomocí definovaných zpráv. Zde byl měřen potřebný čas programu pro generování těchto zpráv v závislosti na jejich množství.



Obr. 5.1: Základní schéma simulace v Python a OMNeT++

Každá simulace byla opakována 10× pro každý počet zpráv, a to konkrétně pro 300; 3 000; 30 000; 300 000 a 3 000 000 zpráv. V případě OMNeT++ je generováno více událostí než je zpráv zaslaných klientem 1, protože jsou zde také přítomny „self“ zprávy a zprávy generované klientem 2 a zásobníkem. Finální výsledky jsou uvedeny níže a v publikaci [A9]:

```
SimPy
Duration:   0.0443761 seconds   300 units
Duration:   0.524 seconds      3000 units
Duration:   4.729 seconds     30000 units
Duration:  46.634 seconds    300000 units
Duration: 477.372 seconds   3000000 units
```

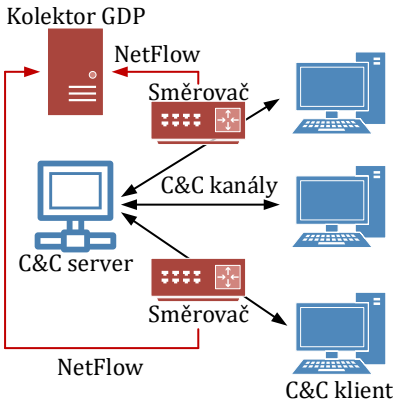
```
OMNeT++
** Event #1477       T=301       Elapsed:  0.002s  (0m 00s)
** Event #14752      T=3001      Elapsed:  0.016s  (0m 00s)
** Event #147502     T=30001     Elapsed:  0.150s  (0m 00s)
** Event #1475002    T=300001    Elapsed:  1.495s  (0m 01s)
** Event #14750002   T=3000001   Elapsed: 14.972s  (0m 14s)
```

Výsledky provedeného testu prokázaly řádově vyšší rychlost zpracování zpráv programem realizovaným v jazyce C++, ale i přes tento závěr byl vybrán jazyk Python pro velice rychlou a snadnou implementaci z vlastního pohledu autora.

6 TESTOVÁNÍ HYPOTÉZY ANALÝZY PŘEŽITÍ

Pro účely ověření hypotézy, že pro každé síťové spojení je definován jiný stav přežití takového provozu a také, že je vyjádřen tento stav jinou křivkou přežití, byl postupně vyvíjen vlastní kolektor NetFlow zpráv. V tomto kolektoru je možné implementovat vlastní algoritmy. O této aplikaci je pojednáno samostatně v kapitole 8 v sekci kolektor 8.1. Výsledky byly publikovány v [A8, A11]. Následující kapitola popisuje implementované kroky pro ověření dané hypotézy. Byly provedeny dva testy pro modelování životního cyklu. Jeden inicializační a druhý podporující danou hypotézu.

V rámci inicializačního testu byl zprovozněn peeringový klient μtorrent na dvou počítačích vzájemně oddělených dvěma ISP (*Internet Service Provider*). Jeden počítač zastupoval server a druhý počítač klienta. Postupně byl stahován referenční soubor o velikosti 100 MB. V rámci druhého testu byl vytvořen vlastní botnet za pomocí jazyka Python. Řídící server navazoval SSH spojení na klientské počítače a zasílal informační příkaz. Řídící kanál byl zprostředkován pomocí Python balíčku `fabric` [65]. Algoritmus periodicky navazoval dané spojení. Tento algoritmus byl inspirován příspěvkem [66]. V tomto případě byla simulována periodická operace. V reálném provozu má takovýto typ komunikace určitou periodicitu výskytu C&C. Tento test byl proveden za účelem zjištění, zda je možné detekovat vztah mezi těmito frekvencemi opakování, pokud nastanou v rozdílné síti a v jiný časový okamžik. Zapojení zařízení pro dané testování je na obrázku 6.1.



Obr. 6.1: Zapojení laboratorních prvků

NetFlow zprávy byly zasílány dvěma Cisco zařízeními v rozdílném adresním rozsahu, ve kterých byl inicializován test. NetFlow zprávy byly zasílány do kolektoru a následně analyzovány. Pro všechny provedené testy a modelování životního cyklu provozu byl použit protokol NetFLow verze 5 a data byla ukládána do vlastního vytvořeného síťového kolektoru. Vyčítání informací o NetFlow bylo zajištěno pomocí databá-

zového konektoru vytvořeného v jazyce Python. V programovém kódu 6.1 je uvedena konstrukce metody databáze.

```
    def survival_read(self, table="Flow", ):
        db = sqlite3.connect("dataset2")
        c = db.cursor()
        sql = """SELECT  IP_SOURCE AS "IP_SOURCE", IP_DEST, (LAST - FIRST) + 1 AS "T",
                        (LAST IS NOT NULL) AS "C" FROM Flow ORDER BY ID DESC"""
        p = pd.read_sql_query(sql, db)
        c.close()
        return p
```

Programový kód 6.1: Metoda selekce údajů z databáze

6.1 Výsledky inicializačního testu

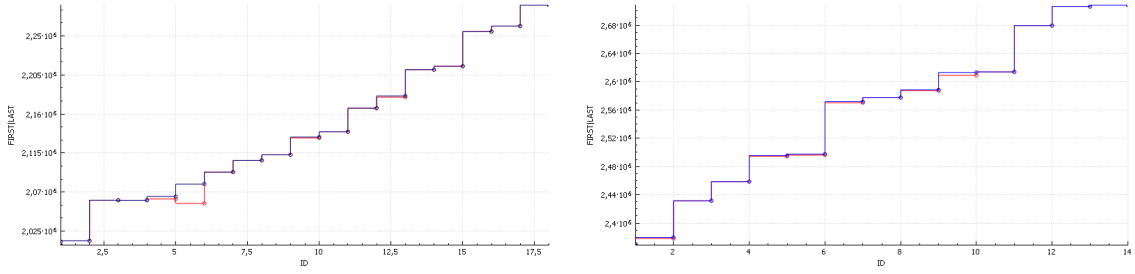
V tomto testu byly vyčteny informace o jednotlivých komunikujících IP adresách a časová UNIX hodnota. Hodnota T značí čas a hodnota C značí cenzuru. V případě IP provozu je uvažována levo cenzurovaná metoda. To je, události nastaly do konce doby pozorování událostí. Níže je uveden výstup ze získaných dat o provozu, tak jak byly předzpracovány pro vstup k vykreslení křivek přežití.

```
          IP_SOURCE          IP_DEST       T  C
0      89.176.9.204     192.168.1.66    1049  1
1      192.168.1.54    192.168.1.255       1  1
2    79.143.185.229     192.168.1.66       1  1
3    115.36.235.169     192.168.1.66       1  1
4     185.130.5.224     192.168.1.66       1  1
5      89.176.9.204     192.168.1.66    1041  1
6      192.168.1.54    192.168.1.255       1  1
7      192.168.1.46    192.168.1.255    1501  1
8      192.168.1.54    192.168.1.255   33009  1
9      192.168.1.54    192.168.1.255       1  1
10     192.168.1.54    192.168.1.255       1  1
11     192.168.1.54     192.168.1.66  106325  1
```

V prvních třech sloupcích je zobrazena zdrojová IP adresa, cílová IP adresa a hodnota t, což je čas kumulovaného trvání pro dané spojení, převzaté z položky UNIX v NetFlow. Tento čas je kalkulován z hodnoty `SysUptime`, kdy byl přijat poslední paket z toku dat s odečtením hodnoty `SysUptime` začátku toku dat. Čtvrtý sloupec představuje hodnotu $delta$. Reprezentuje danou cenzuru. Tato hodnota $delta$ nabývá 1, protože komunikace uzlů byla ukončena během referenčního časového rozmezí. V tomto případě tedy byl provoz ukončen v určeném časovém okně.

Grafy v obrázku 6.2a a 6.2b zobrazují formát z SQL, zachycené toky (ID), poslední čas (červená hodnota) a počáteční čas (modrá hodnota) dvou přenosů 100 MB

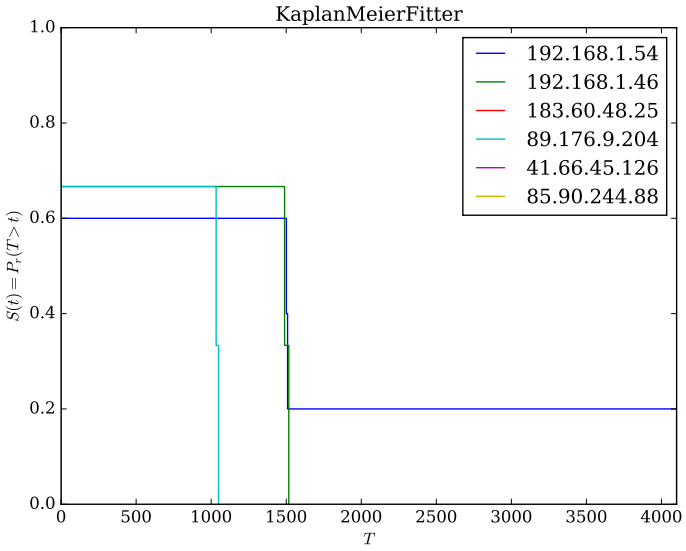
referenčního souboru. Je zde také obsažen i provoz normální. V tomto způsobu reprezentace není zcela možné nalézt vzájemné vztahy provozů.



(a) Dataset 1 (b) Dataset 2

Obr. 6.2: Zobrazení zachycené komunikace v SQL databázi

Získaná data byla podstoupena Kaplan–Meier analýze s levou cenzurou. Výsledek je zobrazen na obrázku 6.3. Komunikace μtorrent klientů a serveru z obou datasetů byla vyfiltrována a v případě tohoto testu byly IP adresy známy předem.



Obr. 6.3: Test Kaplan-Meier

Na ose x je uveden kumulovaný UNIX čas T a na ose y je zobrazena funkce přežití pro každý typ provozu. Po převedení komunikace do tohoto zobrazení je možné vidět souvislost mezi komunikací serveru s oběma klientskými počítači, ač byly pořízeny v jiný časový okamžik a s jinými parametry sítě. V programovém kódu 6.3 je uveden princip provedení takovéto analýzy.

```
kmf = KaplanMeierFitter()
sx = subplot()
def survival_lifeline():
    f = db()
    df = f.survival_read3().head(n=1000000)
    for r in df['IP_SOURCE'].unique():
        ix = df['IP_SOURCE'] == r
        kmf.fit(df['T'].ix[ix], df['C'].ix[ix], label = r)
        kmf.plot(ax=sx, ci_show=False)
survival_lifeline()
```

Programový kód 6.2: Metoda Kaplan-Meier analýzy

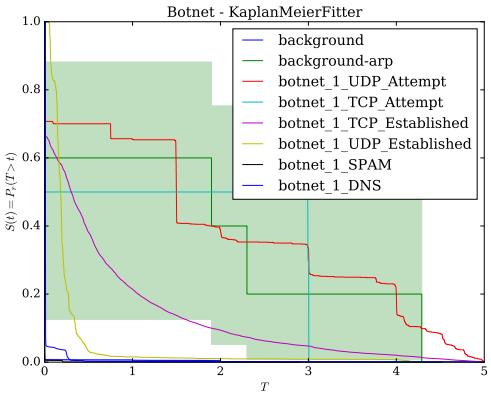
6.2 Výsledky testu podobnosti a modelu provozu

V druhém provedeném testování bylo snahou provést porovnání komunikace jednak zmíněného uměle vytvořeného botnetového serveru a dále převést pomocí této analýzy reálný provoz. Postup analýzy byl shodný jako v případě předchozího testování.

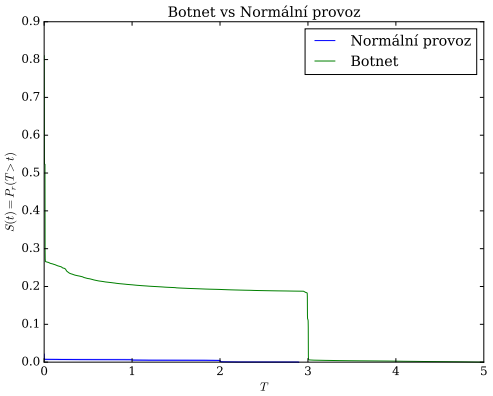
Jako podklad pro data o reálném provozu byl stažen dataset z projektu IPS Stratosphere, konkrétně se jednalo o .csv soubor CTU-Malware-Capture-Botnet-1 [67]. Informace o infikovaném PC: Windows: Win8, IP: 10.0.2.22 (Botnet-V1). Pro testovací účely byl limitován vstup dat pro velikost 1 000 000 položek. Komunikace IP x.22 byla separována a podstoupena analýze. Extrahovaný formát dat je částečně uveden níže:

```
0                   SrcAddr           DstAddr         T C
1         00:00:00:00:00:00 00:00:00:00:00:00  0.000000 0
2                   0.0.0.0         10.0.2.22  2.003218 1
3                        :: ff02::1:ff01:e8a5  0.000000 0
4 fe80::705a:530f:1701:e8a5           ff02::2  4.003125 1
5 fe80::705a:530f:1701:e8a5          ff02::16  0.498083 1
6 fe80::705a:530f:1701:e8a5           ff02::2  0.000000 0
7 fe80::705a:530f:1701:e8a5         ff02::1:2  3.004039 1
8                 10.0.2.22          10.0.2.2  0.000096 1
...                     ...               ...       ... .
999970            10.0.2.22           8.8.8.8  0.010117 1
999971            10.0.2.22           8.8.8.8  0.010097 1
999972            10.0.2.22     94.100.176.20  0.000927 1
999973            10.0.2.22     74.125.142.26  0.001110 1
```

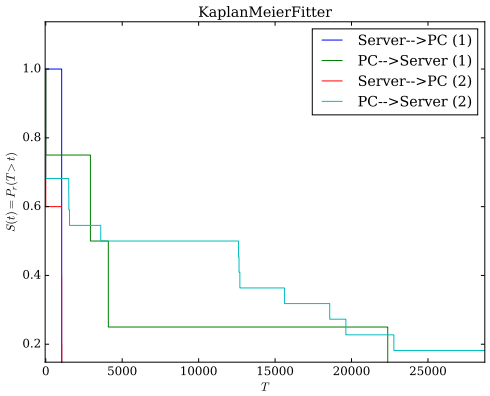
Poté co byl zformován tento dataset, byl podstoupen danému rozboru, jehož výsledek je zobrazen na obrázku 6.4a. V zásadě se dá říci, že osa y reprezentuje pravděpodobnost komunikace v čase T na ose x. V tomto případě byly hledány křivky pro specifické protokoly a služby botnetu, jako je TCP, UDP, SPAM, DNS a pro normální komunikaci. Dohromady byla vytvořena tabulka událostí a medián hodnot, doba trvání a konfidenční interval.



(a) Detailní analýza provozu



(b) Porovnání provozu a botnet



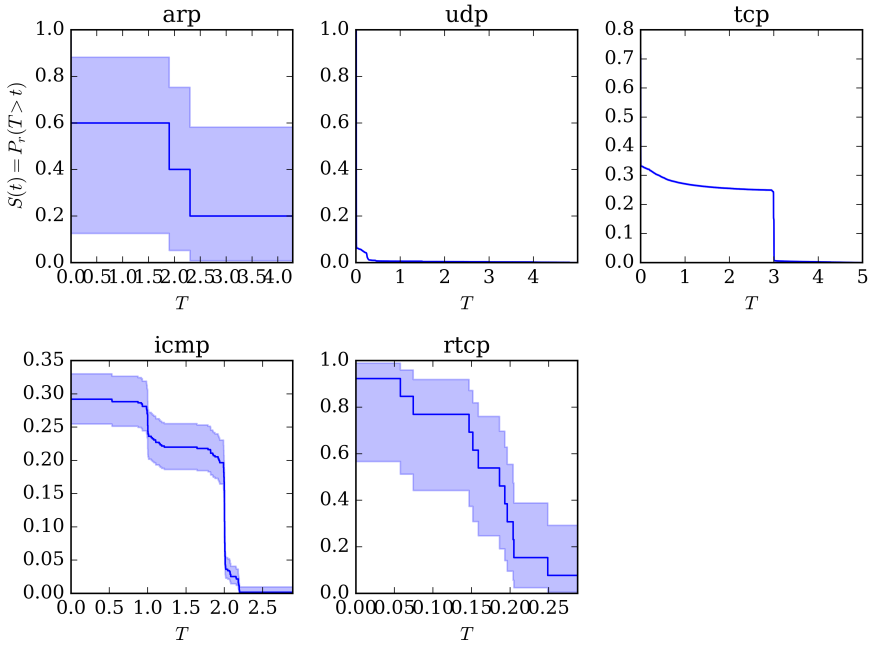
(c) Podobnost C&C zpráv

Obr. 6.4: Test Kaplan-Meier Botnet

```
Medián provozu v pozadí je 0.000436
Medián arp provozu v pozadí je 1.898133
Medián botnet_1_UDP_Attempt je 2.193814
Medián botnet_1_TCP_Attempt je 3.003442
Medián botnet_1_TCP_Established je 0.353073
Medián botnet_1_UDP_Established je 0.180717
Medián botnet_1_SPAM je 0.00136
Medián botnet_1_DNS je 0.010147
```

V grafu na obrázku 6.4b je porovnána křivka přežití pro celý botnet provoz a pro normální provoz.

Graf 6.4c reprezentuje analýzu pro simulovaný provoz C&C zpráv. Predikce ukončení času T je pro komunikaci serveru s počítači shodná, ačkoliv byl tento provoz zachycen v jiném časovém období a pořadí. Dále pak na obrázku 6.5 jsou zobrazeny zvlášť křivky pro jednotlivé protokoly z datasetu.



Obr. 6.5: Zobrazení křivek přežití pro jednotlivé protokoly

Závěrečná zjištění a shrnutí analýzy přežití

Pomocí převedení provozu do zobrazení jejich křivek přežití je možné vyselektovat jednotlivé typy provozu, či vytvořit referenční model. Jejich průběh je závislý na množství zachycených dat a výsledné zobrazení se v zásadě nelišilo od velikosti 10 000 zpráv v sadě. Hodnota $p - value$, hypotézy, že se křivky přežití laboratorního provozu a simulovaných C&C zpráv shodují byla 0, 17090, oproti tomu C&C komunikace $p - value$ dosahovala $\cong$ 0, 75. Pomocí analýzy přežití je možné usuzovat o typu provozu a sestavit referenční mapu provozu.

7 TESTOVÁNÍ EVOLUČNÍCH ALGORITMŮ

Tato část je věnována praktickému testování vlastností genetických algoritmů za účelem ověření teoretických poznatků a vlivu jejich operátorů. Dále také bylo záměrem prakticky naprogramovat genetický algoritmus v jazyce Python bez použití dostupných balíčků třetích stran pro hlubší pochopení jednotlivých principů.

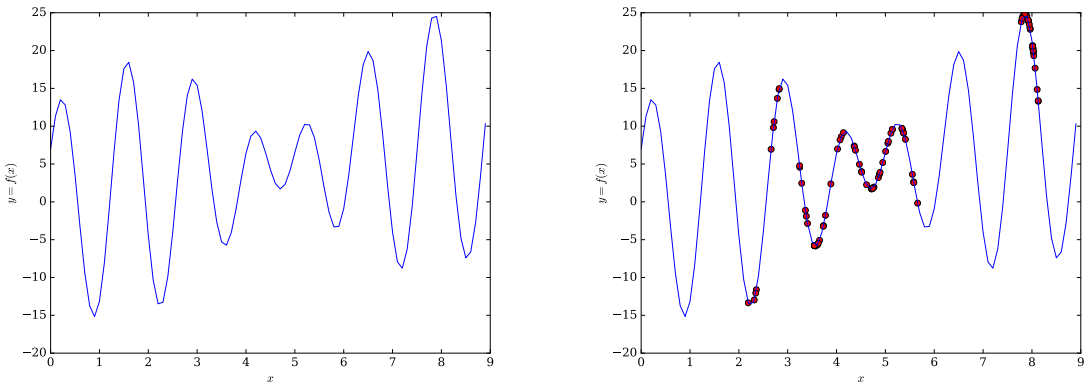
V druhé části testování byly ověřovány možnosti výpočtu referenčních funkcí integrovanými moduly optimalizačních úloh v programu SCILAB, MATLAB a PyCharm Professional. Byly vyzkoušeny i jiné hotové projekty, jako je například GPLAB [68] (*angl. A Genetic Programming Toolbox for MATLAB*) či NGPM [71] (*A NSGA-II Program in Matlab*).

7.1 Ilustrace 1 – Maximalizace dané funkce

Problémem GA a jeho operátorů je uvíznutí v lokálním minimu či maximu řešeného příkladu. V následující ukázce bylo prakticky odzkoušeno, jaký vliv na nalezení řešení má jeho koncepce a použití aritmetického křížení, rozložení a mutace. Zdrojový kód je uveden v příloze. Je použita následující funkce (7.1). Průběh funkce má vnitřní ostrá lokální minima a maxima a představuje vhodnou úlohu k testování maximalizace.

$$y = x + 10sin(5x) + 7cos(4x)\,, \tag{7.1}$$

kde definiční obor hodnot funkce pro hodnotu x je v intervalu $D(x) = \langle 0, 10 \rangle$, kde $x \in \mathbb{R}$. Je hledáno globální maximum funkce f v bodě $M \in D(f)$. Její grafický průběh je na obrázku 7.1a a samotné řešení je vykresleno na obrázku 7.1b.



(a) Ilustrace GA 1 – průběh funkce

(b) Ilustrace GA 1 – řešení

Obr. 7.1: Ilustrace GA 1 – maximalizace

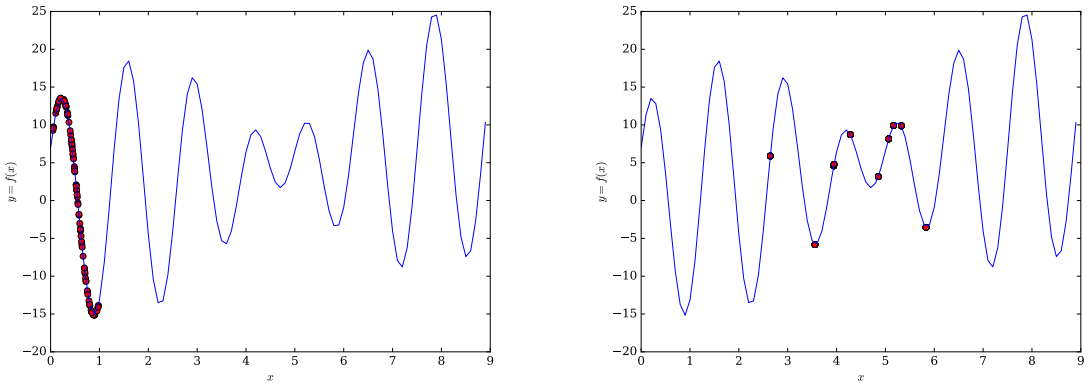
Celá populace je složena z deseti jedinců o velikosti chromozomu tvořeného deseti reálnými čísly, které byly v prvním kroku algoritmu náhodně vygenerovány. Na obrázku 7.1b je vyobrazena hodnota funkce $f(x)$ pro každý gen jedince v desátém kroku algoritmu. Maximální hodnoty bodu $M \in D(f)$ pro každý krok algoritmu je vypsán níže:

```
Vysledek 0 generace, hodnota max f(8.81938905) = 24.716343331
Vysledek 1 generace, hodnota max f(8.78274754018) = 24.716343331
Vysledek 2 generace, hodnota max f(8.78274754018) = 24.716343331
Vysledek 3 generace, hodnota max f(8.63202680329) = 24.716343331
Vysledek 4 generace, hodnota max f(8.63202680329) = 24.8258473787
Vysledek 5 generace, hodnota max f(8.52527666597) = 24.8533714977
Vysledek 6 generace, hodnota max f(8.44129579387) = 24.8533714977
Vysledek 7 generace, hodnota max f(8.36758208221) = 24.8533714977
Vysledek 8 generace, hodnota max f(8.36758208221) = 24.8552057909
Vysledek 9 generace, hodnota max f(8.12830823987) = 24.8552057909
......
Vysledek 27 generace, hodnota max f(7.85673300737) = 24.8553628465
```

Algoritmus postupně nacházel řešení a v 9 kroku vypočítal maximální hodnotu, a to v bodě pro $x = 8.12830823987$ s hodnotou funkce:

$f(8.12830823987) = 24.8552057909$ a od 27 kroku se samotný výpočet algoritmu ustálil na hodnotách s výsledkem:

$f(7.85673300737) = 24.8553628465$. Na následujících dvou obrázcích 7.2a a 7.2b je zobrazen vliv operátorů, mutace a elitismu.



(a) Ilustrace GA 1 – vliv mutace

(b) Ilustrace GA 1 – vliv elitismu

Obr. 7.2: Ilustrace GA 1 – vliv operátorů

V případě přetížení operátoru mutace na hodnoty pravděpodobnosti blízké 1 je vidět uvíznutí algoritmu v lokálním maximu. Ve druhém případě, kdy byl nesprávně nastaven výběr dvou rodičů, bylo vybíráno ze středních hodnot, algoritmus uvízl ve střední hodnotě dané funkce.

7.2 Ilustrace 2 – Výkonnost algoritmu

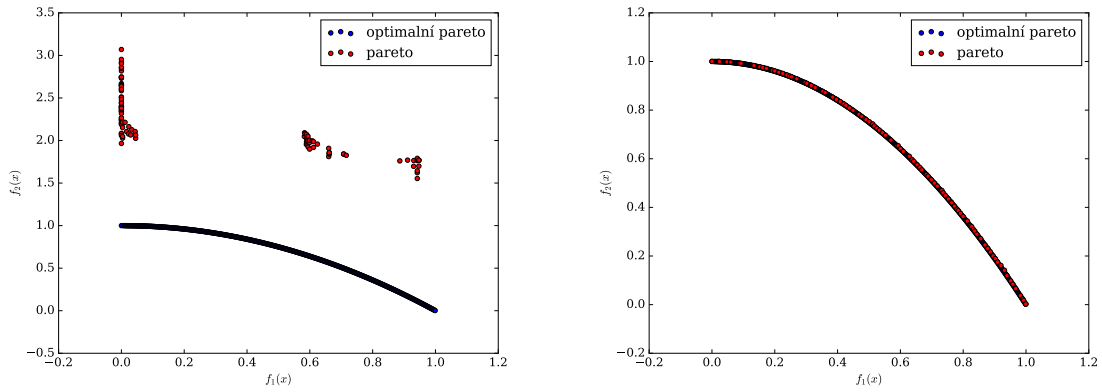
Důležitým krokem pro budoucí hodnocení optimalizačních algoritmů je testování jejich výkonnosti. Pro vlastní testování byl zvolen problém ZDT (*Zitzler-Deb-Thiele's 2*) [69]. Samotný základ ZDT problému má dvě objektivní funkce definované autorem Zitzler a kol. [70] a je tvořen minimalizací dvou funkcí f_1 a f_2 rovnice (7.2).

$$\begin{aligned} f_1(x) &= x_1\,, \\ f_2(x) &= g(x)\; h(f_1(x), g(x)) \end{aligned} \tag{7.2}$$

Celkem jsou definovány ZDT1, ZDT2, ZDT3 a ZDT4 problémy. ZDT2 testovací problém je formulován viz rovnice (7.3).

$$\begin{aligned} f_1 &= x_1\,, \\ f_2 &= g(1 - (x_1/g)^2)\,, \\ g(\mathbf{x}) &= 1 + (9/(n-1)) \textstyle\sum_{i=2}^{n} x_i\,, \end{aligned} \tag{7.3}$$

kde $n = 10$ a $x_i \in [0, 1]$. Pareto region je definován v rozmezí $(0 \leqq x_i \leqq 1)$ optimální pareto fronta $h(f_1, g) = 1 - (f_1/g)^2 = (f_1, 1 - f_1^2)$. Označení pareto optimální je výrazem pro kompromisní množinu řešení, kdy všechny funkce ve více-objektivním algoritmu mají stejnou váhu, či důležitost. Dále byl také zvolen problém ZDT1. Rozdíl mezi těmito dvěma problémy je, že zatímco ZDT1 vrací horizontální vektor ohodnocení víceobjektivních funkcí a představuje kontinuální pareto frontu, ZDT2 představuje kontinuální, nekonvexní pareto frontu. ZDT problémy byly otestovány s genetickým algoritmen NSGAII [71]. Grafické znázornění ZDT2 pareto optimálního řešení a jeho výstup z programu Python s využitím knihovny DEAP je viz obrázek 7.3a, pro velikost populace o 25 jedincích a na obrázku 7.3b pro velikost populace o 250 jedincích.



(a) Pareto DEAP ZDT2 – 25 jedinců

(b) Pareto DEAP ZDT2 – 250 jedinců

Obr. 7.3: Pareto fronta – NSGAII.

Výkonnost dvou víceobjektivních algoritmů je možné měřit více způsoby. Jedním takovým způsobem je kvantitativní měření, které představuje IGD (*Inverted Generational Distance*) autorů Veldhuizena a Lamonta [72], který je definován výrazem (7.4).

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{7.4}$$

V tomto výrazu představuje hodnota n počet všech nedominantních řešení a d_i představuje Euklidovskou vzdálenost mezi každým a nejbližším řešením vzhledem k samotné pareto frontě. Metrika IGD představuje měření mezi pareto frontou a nedominantním řešením.

Výsledky jednotlivých modelování v uvedených programech MATLAB a SCILAB dosahovaly rozdílných výsledků. V průměrném výsledku při použití algoritmu NSGAII pro problém ZDT1 dosahoval výpočet IGD hodnotu 0,4289 a ZDT2 0,7243. Oproti literatuře nebylo dosaženo přesných výsledků, ale i tak se dají tyto výsledky porovnat. Pokud zde konverguje řešení k hodnotě nula, rozprostírají se nedominantní řešení na vlastní pareto frontě.

7.3 Ilustrace 3 – Problém batohu

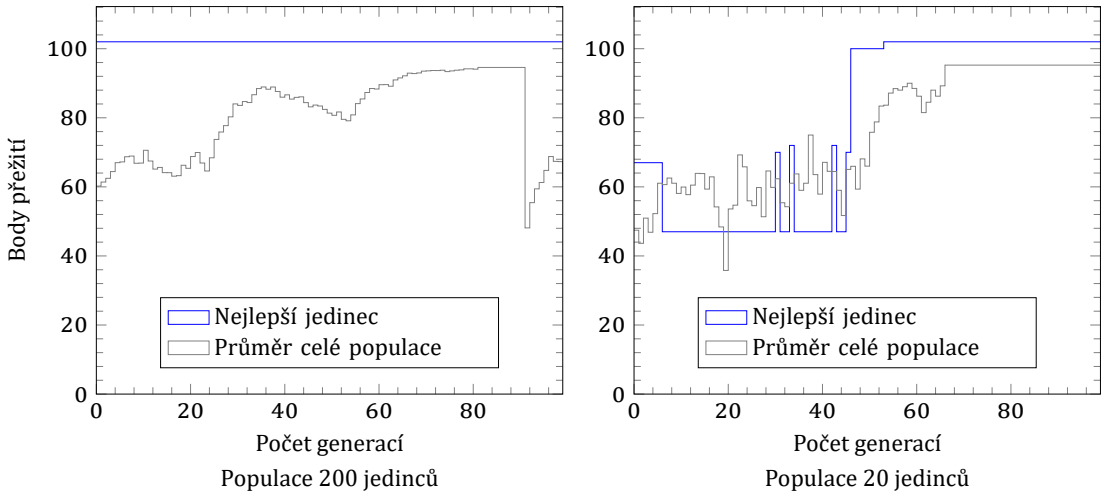
Klasická ukázka funkce genetického algoritmu je na způsobu řešení „problému batohu“ (*Knapsack Problem*), zkráceně KP. Jedná se o kombinatorickou optimalizační úlohu. Tuto úlohu lze řešit vícero způsoby, jako například pomocí dynamického programování. Pro svou povahu NP[6] problému kombinatorické optimalizace se hodí pro řešení pomocí genetických algoritmů.

Batoh zde představuje určitou kapacitu C, váhu, kterou je schopen unést a je zde k dispozici n položek předmětů i, které je možné umístit do tohoto batohu. Každý předmět má určenou váhu C_i a určitý počet bodů přežití V_i. Úlohou je vložit takové předměty, aby byla maximalizována hodnotu bodů přežití, a také zároveň maximalizována hodnota vložené celkové váhy předmětů. Níže ve výpisu je uvedeno základní nastavení navrženého řešení problému a algoritmu. Zdrojový kód je součástí přílohy.

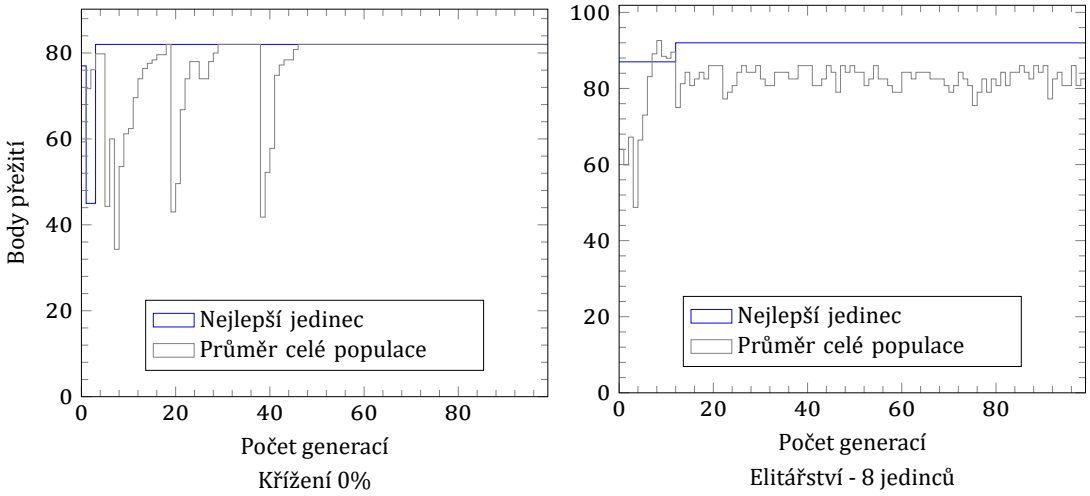
```
MAX_WEIGHT = 20                # Maximální váha
NUMBER_OF_GENERATIONS = 100    # Počet generací
POPULATION_SIZE = 20           # Velikost populace
MUTATE_RATE = 0.01             # Rozsah mutace
CROSSOVER_RATE = 5             # Binární křížení
MAX_ELITE = 2                  # Počet nejlepších jedinců
```

[6]Mezi NP těžké úlohy patří například úloha obchodního cestujícího nebo úloha batohu. Naopak mezi NP úplné úlohy může být uvedena úloha hledání Hamiltonova cyklu. [48]

Na obrázku 7.4 je zobrazen průběh řešení GA v závislosti na velikosti populace 20 a 200 jedinců. Na obrázku 7.5 jsou zobrazeny průběhy GA při manipulaci s genetickými operátory. Při vyloučení křížení populace algoritmus nedosáhl nejlepšího řešení a došlo k jeho uvíznutí. Stejná situace se opakovala také při nevhodné volbě počtu elitních jedinců. Hodnoty nejlepšího možného řešení jsou zobrazeny ve výpisu algoritmu.



Obr. 7.4: Průběh GA – vliv velikosti populace



Obr. 7.5: Průběh GA - vliv parametrů

```
Nejlepsi reseni je:
| [1 1 0 1 1 1 1] | ['kapesni nuz', 'fazole', 'cibule', 'spaci pytel', 'lano', 'kompas']
| vaha: 20 | body: 102 | stredni hodnota: 95.25 | v generaci: 100 |
```

7.3.1 Závěrečná zjištění a shrnutí ilustrací GA

Z prozkoumané problematiky vyplývá, že účelovým funkcím a také genetickým operátorům musí být věnována zvláštní pozornost. Mohou velmi ovlivnit výsledek samotný. Tato problematika je probírána v mnohé literatuře, i v literatuře zde odkazované. Mnoho vědců včetně DeJonga zkoumali parametry operátorů GA a zjistili, že nejlepší výsledky jsou dosahovány při velikosti populace 50, pravděpodobnosti křížení $P_c = 0,6$ a pravděpodobnosti mutace $P_m = 0,001$. Elitářství s výběrem 2 jedinců.

Pro genetické algoritmy platí tedy limitace, jako je vhodnost řešené úlohy touto metodou, frekvence křížení by měla být 80% – 95%. Frekvence mutace by měla být nízká, nepřesahující hodnoty 0,5% – 1%. Metoda selekce by měla vyhovovat danému problému a nejdůležitější částí je kriteriální funkce, která musí být přesná podmínkám řešeného problému.

Existuje mnoho testovacích problémů (*Benchmark*) pro jednotlivé typy reprezentací genetických algoritmů, které zde nebyly uváděny. Tato praktická část posloužila zejména k vytvoření vlastního zdrojového kódu genetického algoritmu. Níže je uveden výběr z dalších testovacích problémů. Jednotlivé testovací problémy jsou uváděny v anglickém jazyce pro dodržení původních názvů od jejich autorů, globální optimum je označeno GO, funkce F. Zdroj DEAP Benchmark [73].

Kontinuální optimalizace:

$$\left.\begin{aligned} F : f(x) &= x_0^2 + 10^6 \sum_{i=1}^{N} x_i^2 , \\ GO : x_i &= 0, \forall i \in \{1 \dots N\}, f(x) = 0 \end{aligned}\right\} \quad \text{Cigar test} \tag{7.5}$$

Více-objektivní optimalizace: Do této oblasti se řadí uvedený problém ZDT2 z kapitoly Ilustrace 2, dále například Fonsecova více-objektivní funkce (7.6).

$$\left.\begin{aligned} F1 : f_1(x) &= 1 - e^{-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2} , \\ F2 : f_1(x) &= f_1(x) = 1 - e^{-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2} \end{aligned}\right\} \quad \text{Fonseca test} \tag{7.6}$$

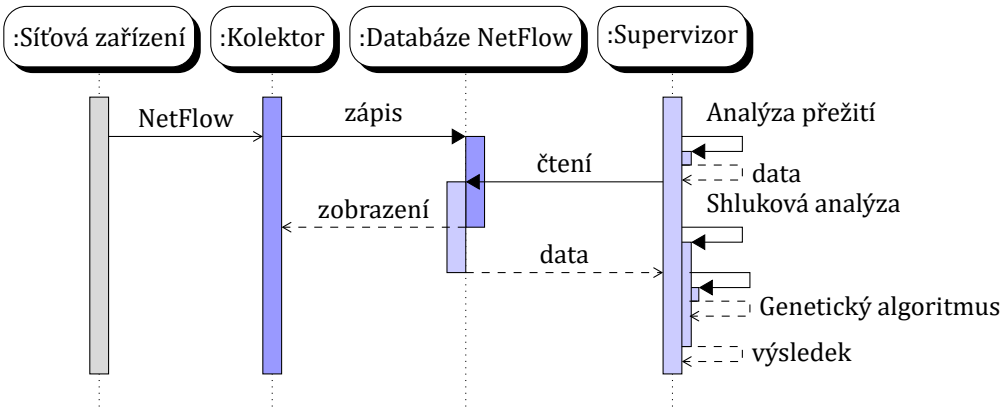
8 NÁVRH ALGORITMU A MODELU

V této sekci je představen vlastní model. Tento model je založený na konceptu detekce anomálií provozu na základě statistické metody s využitím existujících funkcionalit síťových prvků NetFlow. Model využívá genetického algoritmu pro výpočty daných výsledků statistické metody analýzy přežití. Cílem tohoto modelu je schopnost analyzovat provoz v rozdílných časových úsecích a nalézt míru shody doby přežití pro jednotlivá síťová spojení. Takováto shoda dimenzí křivek, či proměnných analýzy přežití potom představuje danou anomálii ve sledovaném provozu.

Vlastní vytvořená aplikace pro účely testování a ověřování výsledků se skládá ze dvou hlavních modulů, a to modulu pro kolekci informací o provozu nazvaného kolektor (*Collector*) a modulu pro vyhodnocení provozu nazvaného supervizor (*Supervisor*). Modul kolektor naslouchá příchozí provoz a ukládá informace z NetFlow zpráv do interní databáze. Dále provádí základní analýzu provozu, konkrétně sumarizaci UNIX času pro jednotlivé komunikující IP adresy.

Modul supervizor je nezávislý na modulu kolektor. Detailní princip je uveden v podkapitole 8.2. Pro každé unikátní spojení jsou z databáze NetFlow vyčteny informace o počtu spojení pro danou IP adresu, trvání každého spojení a o ukončení spojení. Následně jsou vypočteny hodnoty křivek přežití pro tato spojení a pomocí genetického algoritmu jsou data rozdělena do jednotlivých shluků ke zjištění, které provozy mají k sobě nejblíže z pohledu podobnosti křivek přežití.

Pro ověřování metod bylo nutné sestavit vlastní model síťové sondy. Celý funkční model je nazván GDP (Genetic Decision Probe). Základní princip je uveden na obrázku 8.1.



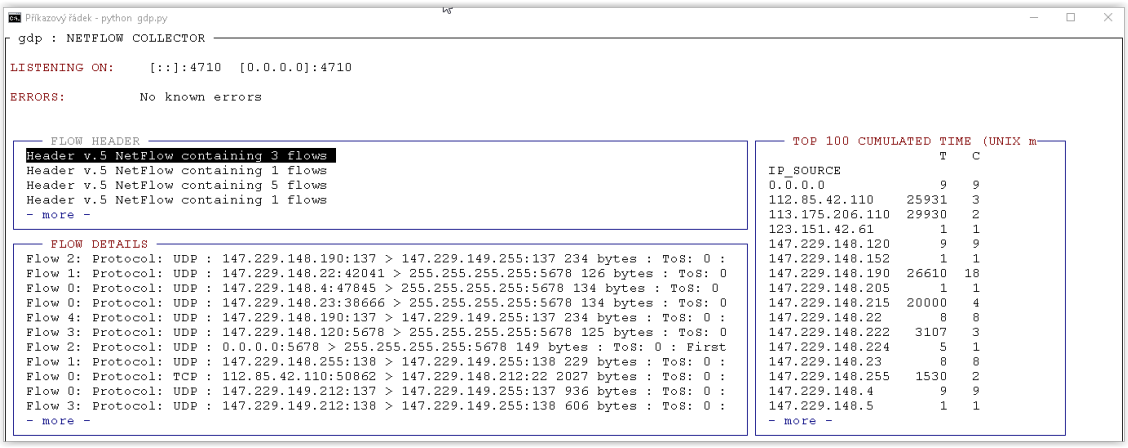
Obr. 8.1: Základní sekvenční schéma GDP

8.1 Modul kolektor

Koncept kolektoru je založen na schopnosti zpracovat NetFlow zprávy verze 1 a 5. Modul kolektor se skládá ze dvou hlavních částí. Hlavní spouštěcí program je definován v souboru <gdp.py>. Dalšími částmi programu jsou <interfaces>, <core>, <database> a <common>, které obsahují jeho dílčí operační části.

8.1.1 Uživatelské rozhraní

Uživatelské rozhraní TUI (*Terminal User Interface*) je vytvořen pomocí dostupného balíčku npyscreen a poskytuje vše potřebné pro interakci s uživatelem za použití systémové konzoly, viz obrázek 8.2.



Obr. 8.2: Grafické zobrazení aplikace

Exportovat NetFlow zprávy je možné buď z fyzického portu nebo z podružného portu zvaného „subinterface“. Pro export z fyzického portu je uvedena konfigurace laboratorního směrovače níže.

```
ip cef
!
ip flow-export version 5
ip flow-export destination 147.229.148.215 4710
!
interface FastEthernet0/0
 ip route-cache flow

router#show ip flow export
Flow export v5 is enabled for main cache
  Exporting flows to 147.229.148.215 (4710)
  Exporting using source IP address 147.229.148.212
  Version 5 flow records
  1073024 flows exported in 310851 udp datagrams
  ...
```

Kolektor naslouchá příchozí NetFlow zprávy na portu číslo <4710>. Tento port a další nastavení může být změněno v souboru <config.txt>. Celý soket[6] je tvořen všemi dostupnými síťovými rozhraními zařízení, na kterém je program spuštěn. V části FLOW HEADER jsou zobrazovány informace vyčítané z hlaviček přichozích NetFlow zpráv. Ve FLOW DETAILS jsou poté vypisovány detaily o každém toku, jako je: číslo, zdrojová IP adresa, cílová IP adresa, typ služby a první a poslední čas zaznamenaného provozu v UNIX formátu.

Průběžná analýza síťového provozu je potom zobrazována v části TOP. Tato analýza představuje sumarizaci celkového času trvání spojení a počet pozorování spojení pro každou IP adresu. Tato statistická analýza používá informace z databáze, kam jsou všechna NetFlow zapisována. Formát databázových polí kopíruje formát NetFlow. Databázový soubor je tvořen sloupci ID, SYS_UPTIME, UNIXS, FIRST, LAST, IP_SOURCE, IP_DEST, PROTO, SOURCE_PROTO, DEST_PROTO a timestamp.

Vytvořený program automaticky ukládá informace do databázového souboru s názvem <dataset.sqlite3>. Informace obsažené v uvedené databázi jsou automaticky smazány při startu programu. Uchovat tyto informace je možné změnou konfigurace v konfiguračním souboru, a to přepsáním hodnoty <YES> na hodnotu <NO>. Historická data nebudou potom smazána. Funkce databázového konektoru je uvedena v programovém kódu 8.1.

Byl také použit balíček threading, protože program běží ve dvou nezávislých vláknech a s jedním generálním zámkem. Nejprve je spuštěno vlákno konzolového prostředí, a poté je pro vytvoření soketu spuštěno druhé vlákno programu. Tímto je dosaženo spuštění dvou nezávislých smyček, protože vytvořený soket nepřetržitě naslouchá příchozímu provozu a paralelně je aktualizováno uživatelské rozhraní.

```
def survival_read(self, table="PFlow", ):
  db = sqlite3.connect(self.database_name)
  cursor = db.cursor()
  sql = """SELECT IP_SOURCE AS "IP_SOURCE", IP_DEST,
      DURATION AS "T",(DURATION IS NOT NULL AS "C"
              FROM PFlow ORDER BY ID DESC"""
  data = pd.read_sql_query(sql, db)
  cursor.close()
  return data
```

Programový kód 8.1: Metoda selekce údajů z databáze

Jak bylo uvedeno, součástí konzolového prostředí je analýza trvání jednotlivých spojení. Informace pro tuto analýzu jsou opět získána z databáze. Hodnoty jsou vypočítávány z hodnoty <SysUptime> v NetFlow z poslední kolekce hlášení o provozu. Zde je možné sledovat statistiku nejdéle trvajícího spojení a počet takto navázaných spojení

[6] V IP sítích je soket tvořen dvojicí, IP adresou a číslem portu.

v téměř reálném čase. Metoda časové analýzy kolektoru je uvedena v programovém kódu 8.2.

Metoda <read_sql_query> z balíčku pandas čte data z databáze a potom jsou tato data předpřipravena pomocí hodnoty <head>, kde je vyčteno 100 vstupů IP adres. Dále jsou tato data seskupena podle těchto IP adres do jednotlivých skupin a informace jsou zpracovány metodou <survival_max_time_per_ip()>, která vrací sumu agregovaných hodnot.

```
import warnings
import numpy as np
from database import Database as db

def survival_max_time_per_ip():
    with warnings.catch_warnings() as s:
        warnings.filterwarnings("error")

        try:
            f = db("./database/dataset.sqlite3")
            df = f.survival_read()
            daf = df.head(n=100)
            grouped = daf.groupby("IP_SOURCE")
            return "%s" %
      (grouped.aggregate(np.sum))
        except RuntimeWarning:
            return s
```

Programový kód 8.2: Metoda časové analýzy v GDP

8.2 Modul supervizor

V této kapitole je popsán princip modulu supervizor a jeho návrh, včetně jednotlivých postupů a návrhu algoritmů. Dále jsou zde uvedeny postupy a výsledky provedeného testování implementace navržených algoritmů a jejich vzájemné porovnání.

8.2.1 Výčet dat z databáze a návrh implementace analýzy přežití

Modul supervizor načítá v průběžné smyčce data z databáze NetFlow kolektoru. Pro jednotlivá spojení *SD* (*Source to Destination*), ze sondy *P* jsou vypočteny hodnoty

přežití $S(t) = Pr(T > t)$ pro individuální časovou periodu (8.1).

$$\hat{S}(t)_{SD} = \prod_{j=1}^{k} \left(\frac{n_j - d_j}{n_j} \right) \forall SD_P \,,$$
$$S(t)_{IP} = \frac{\text{počet výskytů - počet ukončení}}{\text{počet výskytů}} \,, \tag{8.1}$$

kde čas výskytu události T je reprezentován výpočtem z hodnot `(LAST - FIRST) + 1` z NetFlow toku SD a počet ukončení, resp. cenzurování je reprezentováno volbou 0 nebo 1 databázovým dotazem `case when (LAST - FIRST)>0 then 1 else 0`. Na obrázku 8.3 jsou zobrazeny jednotlivé položky v databázi.

	ID	SYS_UPTIME	UNIXS	FIRST	LAST	IP_SOURCE	IP_DEST	timestamp
1	1	1025904	1452788763	999780	999780	0.0.0.0	255.255.255.255	1452789095.825948
2	2	1037904	1452788775	1010992	1010992	192.168.1.46	192.168.1.255	1452789107.825429
3	3	1074904	1452788812	1036236	1048876	192.168.1.54	192.168.1.255	1452789144.830086
4	4	1074904	1452788812	1047036	1059644	192.168.1.54	192.168.1.66	1452789144.970734
5	5	1087904	1452788825	1061036	1061036	107.150.98.128	192.168.1.66	1452789157.834082
6	6	1134904	1452788872	1095876	1108572	192.168.1.54	192.168.1.255	1452789204.820278
7	7	1160916	1452788898	1134364	1134364	71.6.167.142	192.168.1.66	1452789230.831699
8	8	1191904	1452788929	1164304	1165816	192.168.1.46	192.168.1.255	1452789261.833827
9	9	1191904	1452788929	1175088	1176596	192.168.1.54	192.168.1.255	1452789262.00568
10	10	1239904	1452788977	1213500	1213500	192.168.1.54	192.168.1.255	1452789309.833206
11	11	1253904	1452788991	1208976	1227548	192.168.1.54	192.168.1.255	1452789323.841606
12	12	1286904	1452789024	1195432	1260764	192.168.1.54	192.168.1.66	1452789356.837678
13	13	1286904	1452789024	1256836	1261424	213.57.65.183	192.168.1.66	1452789356.978287
14	14	1286904	1452789024	1267048	1267048	192.168.1.46	192.168.1.255	1452789357.072019

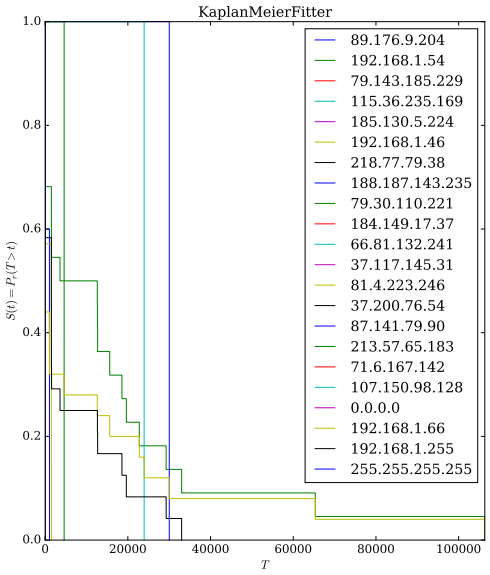
Obr. 8.3: Rozložení databáze NetFlow

Vztah mezi počtem uzlů sítě a možnosti detekce shodného průběhu spojení ovlivňuje počet zapojených síťových sond. Parametrem je bod zvratu, po jehož dosažení možnost rozlišení klesá. Tento parametr je důležitým pro selekci zpráv od jednotlivých uzlů sítě vysílajících NetFlow zprávy. Vycházíme–li ze základní definice o síťové tomografii, viz rovnice (8.2), potom je možné určit takovýto bod pomocí singulárního rozkladu matice síťových zařízení, která zasílají NetFlow toky.

$$\boldsymbol{\nu} = \mathbf{M}\boldsymbol{\varepsilon} \;\; ; \qquad \begin{bmatrix} \nu_1 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}, \tag{8.2}$$

$$\begin{aligned} SVD(\mathbf{A}) &= USV^T \,, \\ QR &= (U_i)^T P_i \,, \\ \mathbf{A}_{\text{Nová}} &= (P_i)^T P_i \end{aligned} \tag{8.3}$$

Vytvořená databáze je rozšířena o hodnotu `engine_id`, která indikuje, ze kterého zařízení byly NetFlow informace zaslány pro možnosti ověření bodu zvratu. Například



Obr. 8.4: Analýza přežití pro všechna SD testovaného provozu

pro provoz 192.168.1.66 je výstupem křivka přežití viz hodnoty níže a obrázek 8.4 pro všechny provozy v daném časovém období. Protože ne všechny provozy mají stejnou délku časové řady, jsou přepočítány do lineárního prostoru o potřebném počtu časových úseků.

```
          192.168.1.66                  192.168.1.66
timeline                  linear space (0, 40000, 10)
0            1.000000     0.000000          1.000000
1            1.000000     4444.444444       0.727273
1033         0.909091     8888.888889       0.636364
1041         0.818182     13333.333333      0.545455
1049         0.727273     17777.777778      0.454545
4589         0.636364     22222.222222      0.454545
12609        0.545455     26666.666667      0.272727
15613        0.454545     31111.111111      0.181818
22769        0.363636     35555.555556      0.181818
23941        0.272727     40000.000000      0.181818
30017        0.181818
65333        0.090909
106325       0.000000
```

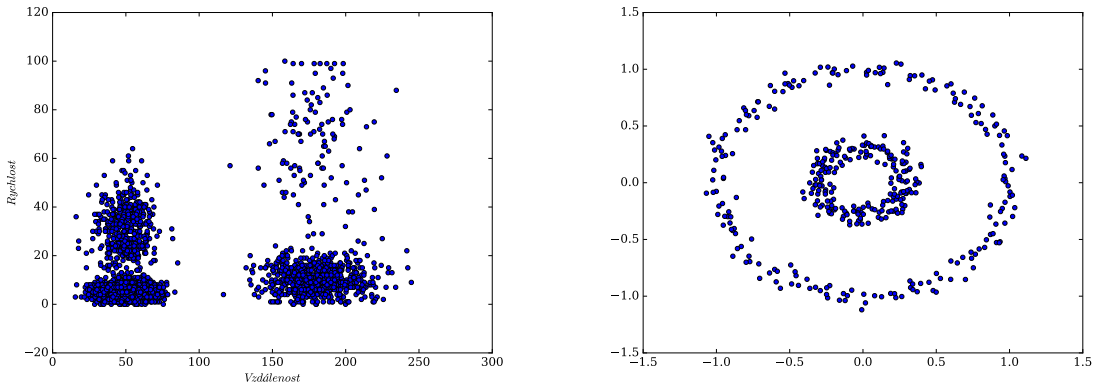
Genová expresní matice je potom vyjádřena $GEM_{(m,n)} = m \times n$ pro časové okno $w(t)$, kde m představuje sadu řešení a n hodnoty $m = \{n1, n2, \ldots, n(t)\}$ sady m v daném lineárním prostoru. Formát zápisu v programu je následující:

```
192.168.1.66  1.00  0.73  0.64  0.55  0.45  0.45  0.27  0.18  0.18  0.18
192.168.1.255 1.00  0.42  0.42  0.28  0.28  0.14  0.14  0.07  0.00  0.00
89.176.9.204  1.00  1.00  0.66  0.33  0.32  0.00  0.00  0.00  0.00  0.00
...
```

8.2.2 Shluková analýza a genetický algoritmus

Získané hodnoty z analýzy přežití jsou podrobeny shlukové analýze pro účely zjištění podobnosti jednotlivých křivek přežití. V této podkapitole je uvedeno porovnání algoritmu K-průměrů (*K-Means*) a navrženého genetického algoritmu.

Pro účely testování vlastních algoritmů byla nejprve zvolena reálná testovací sada řidičů [85], kde byl sledován průměrný počet ujetých kilometrů v závislosti na průměrné rychlosti. Tato sada obsahuje 4 000 hodnot. Dále byl také zvolen testovací vzorek `make_circles` z balíčku `scipy python`. Na obrázku 8.5 je vyobrazeno jejich rozložení ve dvou dimenzionálním prostoru x, y.



(a) Dataset řidiči

(b) Dataset kruhy

Obr. 8.5: Zobrazení vstupních testovacích sad pro shlukovou analýzu

Pro porovnání výsledků je použita časová hodnota běhu algoritmů a Daviesův - Bouldinův validační index (DB). V případě genetického algoritmu je také zjišťována průměrná hodnota Euklidovské vzdálenosti mezi prvky obsaženými ve shluku. Pro potřeby ohodnocení výstupů byla vytvořena třída `GetDaviesBouldin`, obrázek 8.6, vracející hodnotu DB indexu a vnitroshlukové vzdálenosti.

`GetDaviesBouldin(object)`

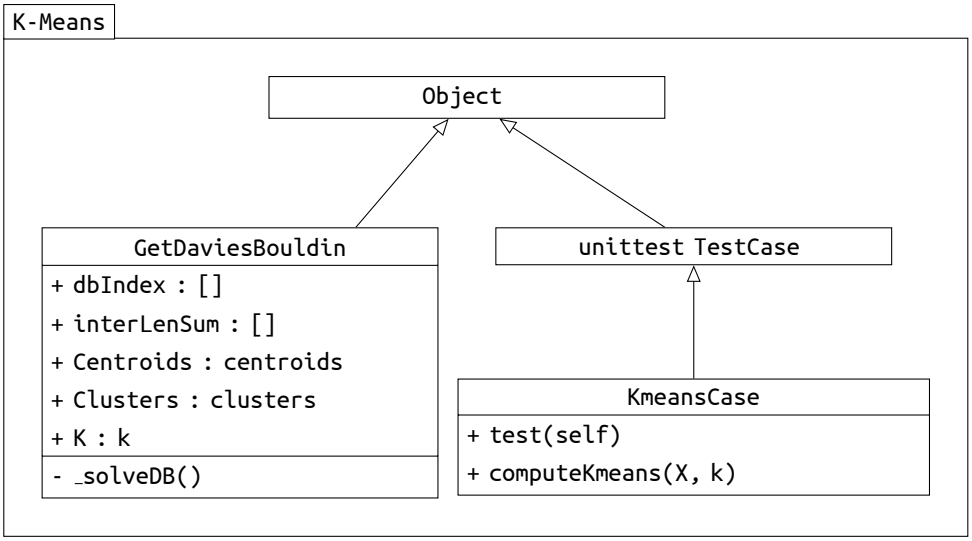
`+ dbIndex : []`
`+ interLenSum : []`
`+ Centroids : centroids`
`+ Clusters : clusters`
`+ K : k`

`- _solveDB()`

Obr. 8.6: Třída Davies-Bouldin index

8.2.3 Implementace algoritmu K-průměrů

Algoritmus K-průměrů je sestaven podle probrané teoretické části. Je použit princip Lloydova algoritmu. Klíčovým krokem jsou průměry $x \in X$, které se rovnají argumentu $\min_{c \in \mathbb{R}^d} \sum_{x \in X} \|c - x\|^2$. Pracuje ovšem pouze se vzdáleností $\mathbf{d}(x, c) = \|x - c\|$. Pomalejší proces, ale odstraňující předešlé omezení je zvolení centroidu c_i dle $\{x \in X | \phi_C(x) = c_i\}$. Třídní diagram K-průměrů je uveden na obrázku 8.7.



Obr. 8.7: Třídní diagram K-Means.py

V prvním kroku jsou zvoleny náhodně $c_i = x_i \in X$. Dále je proveden výpočet Euklidovské vzdálenosti člena x_i vůči centroidu c_i a tento člen je vložen do nejvíce vyhovující skupiny. Centroidy jsou přepočítány a nahrazeny průměrnou hodnotou hodnosti členů shluku. Proces se opakuje dokud nejsou přiřazeni všichni členové. Celý proces je uveden v programovém kódu 8.3.

```
centers = [X[i] for i in random.sample(range(len(X)), k)]

for individum in X:
    for i, centroid in enumerate(centroids):
        if (euclidianDistanceMean(individum, centroid)) <= min(euclidianDistanceMean(individum, \
                            centroid) for centroid in centroids):
            clusters[i].append(individum)
            centroid.clear()
            centroid.append(np.mean(clusters[i], axis=0))
```

Programový kód 8.3: K-Means: přiřazení centroidu a členů

V následujícím kroku konvergence algoritmus porovnává každého člena s vlastním shlukem a shlukem sousedním z pohledu Euklidovské vzdálenosti. V případě nalezení vhodnějšího shluku (klasteru) vybraného člena přesune do tohoto shluku. Tento

krok se stále opakuje, dokud nenastane nulový počet přesunů členů. Klíčové části algoritmu Euklidovské vzdálenosti jsou uvedeny v programovém kódu 8.4 a 8.5.

```
def euclidianDistance(adept, centroids):
        distance = (math.sqrt(sum([(a - b) ** 2 for a, b in zip(adept, centroids)])))
        return distance
```

Programový kód 8.4: K-Means: Euklidovská vzdálenost

```
        relocation = 1
        R = 0
        while relocation != 0:
            R += 1
            for cluster_index, cluster in enumerate(clusters):
                relocation = 0
                for index, get_value in enumerate(cluster):
                    if (euclidianDistanceMean(get_value, cluster)) > (min(euclidianDistanceMean \
                                (get_value, old_cluster) for old_cluster in clusters)):
                        a, value = min(enumerate(clusters), key=lambda x: euclidianDistanceMean \
                                                                (get_value, x[1]))
                        relocation += 1
                        for position, value in enumerate(cluster):
                            if (get_value == value).all():
                                clusters[cluster_index].pop(position)
                        clusters[a].append(get_value)
```

Programový kód 8.5: K-Means: konvergence

8.2.4 Návrh a implementace genetického algoritmu

Cílem v této části bylo vytvořit vhodný genetický algoritmus pro úlohu shlukování dat pro využití v problému nalezení příbuzných křivek z analýzy přežití a zvýšit efektivitu této úlohy. Původně zamýšlený postup využití minimalizace kostry grafu jednotlivých sad m s ohodnocením hran Euklidovskou vzdáleností byl nahrazen shlukovou analýzou. Ta je schopna shlukovat různé typy sad oproti uvažovanému postupu využití grafu, kde by byla nalezena cesta pouze nejkratších vzdáleností.

Genetický algoritmus pracuje s chromozomy řešení daného problému. V tomto případě byl zvolen chromozom obsahující k genů pro k centroidů c_i shluků S. Chromozóm, neboli jedinec α a následně populace P je v prvním kroku sestavena dle výrazu (8.4). Centroid $\mathbf{c}_i = \mathbf{x}_i$ a $\mathbf{x}_i$ představuje vektor hodnot $\mathbf{x}_i = \{n_1, n_2, \dots, n_M\}$, kde n představuje prvek sady M z genové matice.

$$\begin{aligned} \alpha_i = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k) \in C | \mathbf{c}_k = \mathbf{c}_i \in X\,, \\ P = \{\alpha_1, \alpha_2, \dots, \alpha_S\} \end{aligned} \tag{8.4}$$

Chromozom udržuje jen centroidy shluků a při volání ohodnocení jedince jsou přidruženy k těmto centroidům hodnoty $\mathbf{x}_i$. Pro ohodnocení jedince je využito Euklidovské vzdálenosti a Daviesův-Bouldinův validační index. Genetický algoritmus minimalizuje sumu Euklidovské vzdálenosti pro všechny jednotlivé shluky. Výraz (8.5) je účelovou funkcí $f(\alpha)$ daného problému.

$$\text{minimalizace} \sum_{k=1}^{k} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \mathbf{c}_k\|^2, \qquad \text{vzhledem k: } \mathbf{c}_k, S_k \tag{8.5}$$

Kriteriální funkce, tj. ohodnocení jedince (chromozomu) je prováděno na základě sumy průměrných vzdáleností uvnitř shluků, tj. $f(\alpha) \wedge fit_1(\alpha)$ a $\forall \alpha_i \in P : 0 < f(\alpha)_1 < f(\alpha)_2$. Například pro počet shluků $S = 4, k = 4$, jedinec obsahuje 4 geny, každý gen reprezentuje řešení jednoho shluku, potom dostáváme formát řešení:

```
        for i, centroid in enumerate(self.Centroids):
            centroid.clear()
            centroid.append(np.mean(self.Clusters[i], axis=0))

        temp = []
        for i, cluster in enumerate(self.Clusters):
            for index in cluster:
                temp.append(_euclidianDistanceMean(index, self.Centroids[i]))
            inter_lenght[i].append(np.mean(temp[:], axis=0))
```

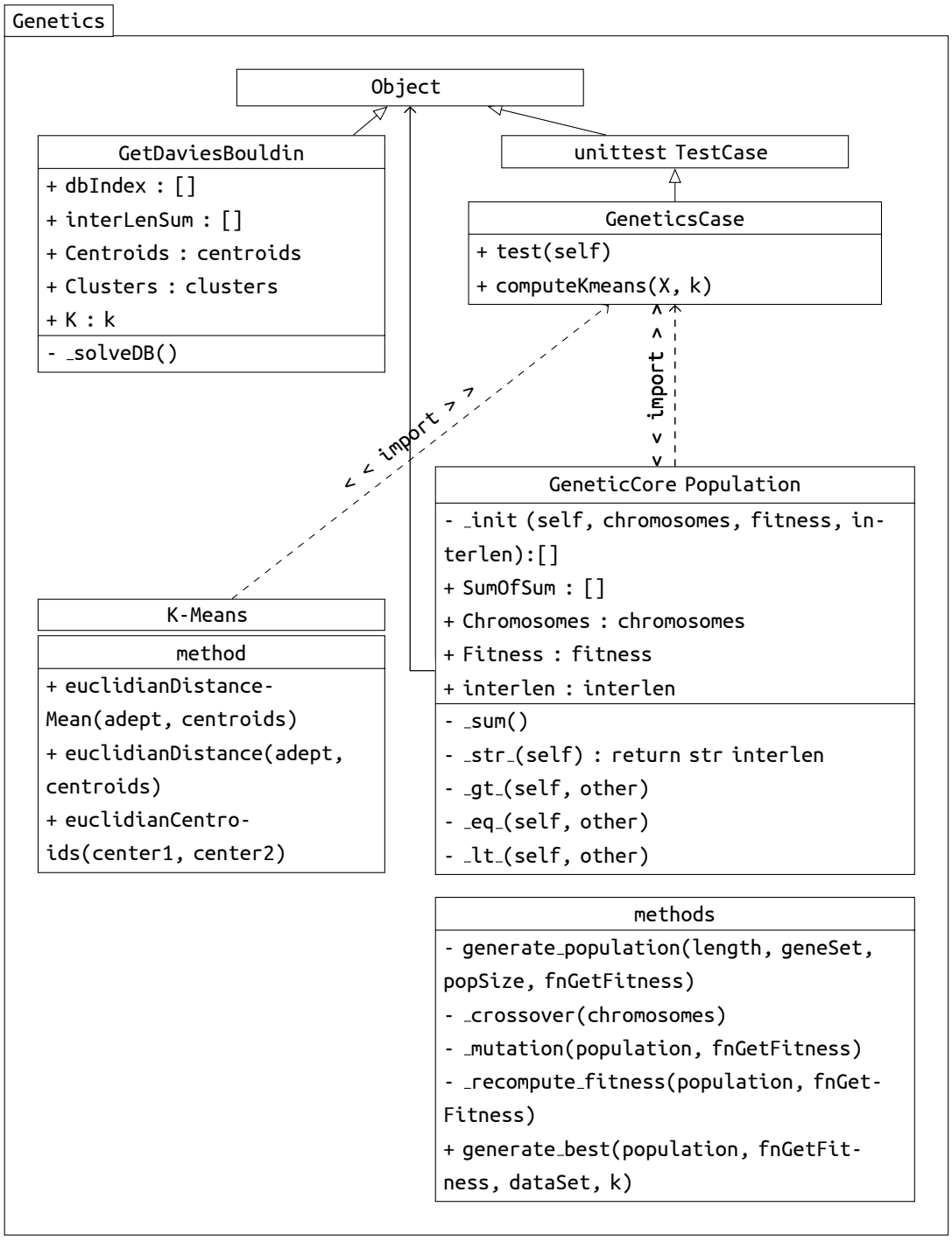
Programový kód 8.6: Kriteriální funkce GA shluků

```
Průměrná vzdálenost uvnitř shluků            Suma:
[[0.0], [0.0], [0.0], [1.5871699711831522]]  1.5871699711831522
```

Potom genetický algoritmus volí jedince s nejnižší hodnotou sumy průměrných vzdáleností v populaci P. Pro zjednodušení běhu algoritmu je volána tato metoda z importované vlastní třídy `GetDaviesBouldin` z předchozího řešení K-průměrů. Celá koncepce kódu genetického algoritmu je uvedena na obrázku 8.8. Tato třída zajišťuje i výpočet Davies-Bouldinova validačního indexu a vrací její hodnotu v proměnné `dbIndex`. Tento validační index představuje druhou kriteriální funkci (8.6).

$$fit_2(\alpha) = DB_\alpha = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left\{ \frac{S_n(Q_i) + S_n(Q_j)}{S_n(Q_i, Q_j)} \right\}, \qquad \forall \alpha_i \in P : 0 < f(\alpha)_1 < f(\alpha)_2 \tag{8.6}$$

Třída `Population` udržuje aktuální hodnoty populace `self.Chromosomes` a ohodnocení jedinců v `self.Fitness` a `self.interlen`. Obsahuje vlastní metody pro porovnání populací `OBJEKT.__lt__(self, other)` a dále `eq` a `gt`.



Obr. 8.8: Třídní diagram GeneticsCore.py

Vždy mezi dvěma populacemi je tedy hledána hodnota minima vnitro-shlukové vzdálenosti. Jedna populace představuje původní populaci a druhá populace představuje upravenou o křížení a mutaci s přepočítanými hodnotami kriteriálních funkcí.

Je tedy prováděno ohodnocení celých populací v rámci provedení elitismu. A to z toho důvodu, že je prováděno křížení jak vertikální, tak také horizontální mezi

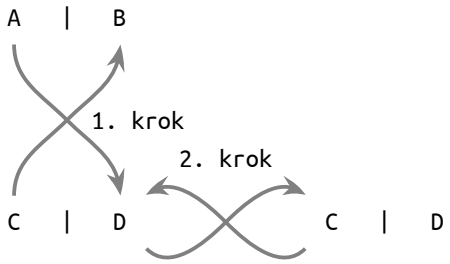
jednotlivými jedinci v populacích. Celý proces je uveden v programovém kódu 8.7. Chybné výsledky (inf, null) po dělení nulou a sama sebou jsou filtrovány pomocí metody `np.ma.masked_invalid` z balíčku numpy python. Tím je zajištěno, že nebudou vybráni chybní jedinci. V uvedeném příkladu je nastaven cyklus opakování `for` na hodnotu 2 cyklů. V každém cyklu je provedeno křížení, mutace a přepočítání hodnoty fitness. Poté je provedeno porovnání populací a vrácena ta populace, která vyhovuje podmínce výběru nejlepšího řešení.

```
def generate_best(population, fnGetFitness, dataSet, k):
    pop = population
    for i in range(2):
        pop.Chromosomes = _crossover(pop.Chromosomes)
        newPop = _recompute_fitness(pop, fnGetFitness)
        newPop = _mutation(newPop, fnGetFitness)
        if pop > newPop:
            pop = newPop
        value, index = (min([(np.ma.masked_invalid(v), i) for i, v in enumerate(pop.interlen)]))
    return pop
```

Programový kód 8.7: Běh algoritmu GA shlukové analýzy

Křížení

V navrženém řešení je vytvořena metoda křížení na základě náhodného výběru jak ve vlastním chromozomu jednotlivých genů, tak také mezi chromozomy, viz obrázek 8.9. V prvním kroku jsou prokříženy vybrané části mezi chromozomy a v druhém kroku pak změněny pozice genů v chromozomech. Tímto způsobem je zajištěno omezení uvíznutí řešení v lokálním minimu hodnot.



Obr. 8.9: Princip křížení chromozomů

Metoda křížení je uvedena v programovém kódu 8.8. Do této metody vstupují pouze proměnné chromozomu ze třídy `Population` a jsou následně vráceny do stejné instance třídy. Toto křížení probíhá vždy na počátku každého cyklu `for` volané metody `generate_best`.

```
def _crossover(chromosomes):
    if len(chromosomes[:]) < 2:
        return chromosomes
    offspring1, offspring2 = random.sample(range(len(chromosomes[:])), 2)
    chromosomes[offspring1], chromosomes[offspring2] = \
        chromosomes[offspring2], chromosomes[offspring1]
    for gene in chromosomes:
        if len(gene[:]) < 2:
            return gene
        offspring1, offspring2 = random.sample(range(len(gene[:])), 2)
        gene[offspring1], gene[offspring2] = \
            gene[offspring2], gene[offspring1]
    return chromosomes
```

Programový kód 8.8: Křížení populace v GDP

Mutace

Mutace daného řešení zajišťuje, aby jednak nedošlo uvíznutí v lokálním a globálním minimu. Dále jsou také pomocí této mutace náhodně nahrazeny stávající hodnoty centroidů, protože v prvním kroku algoritmu je centroid $\mathbf{c}_i$ roven $\mathbf{x}_i$. V této části prováděné mutace je využito třídy `GetDaviesBouldin`, která navrací průměr hodnot $\mathbf{x}_i \in S(k)$. Potom je chromozom tvořen (8.7).

$$\begin{aligned} \alpha_{i:\text{nové}} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k) \in C | \mathbf{c}_k = \emptyset S(k)\,, \\ P = \{\alpha_1, \alpha_2, \ldots, \alpha_S\}\,, \\ \mathbf{c}_{k:\text{nové}} = \mathbf{c}_k = \mathbf{x}_i | i, k : \Omega \longrightarrow (1, N) \end{aligned} \tag{8.7}$$

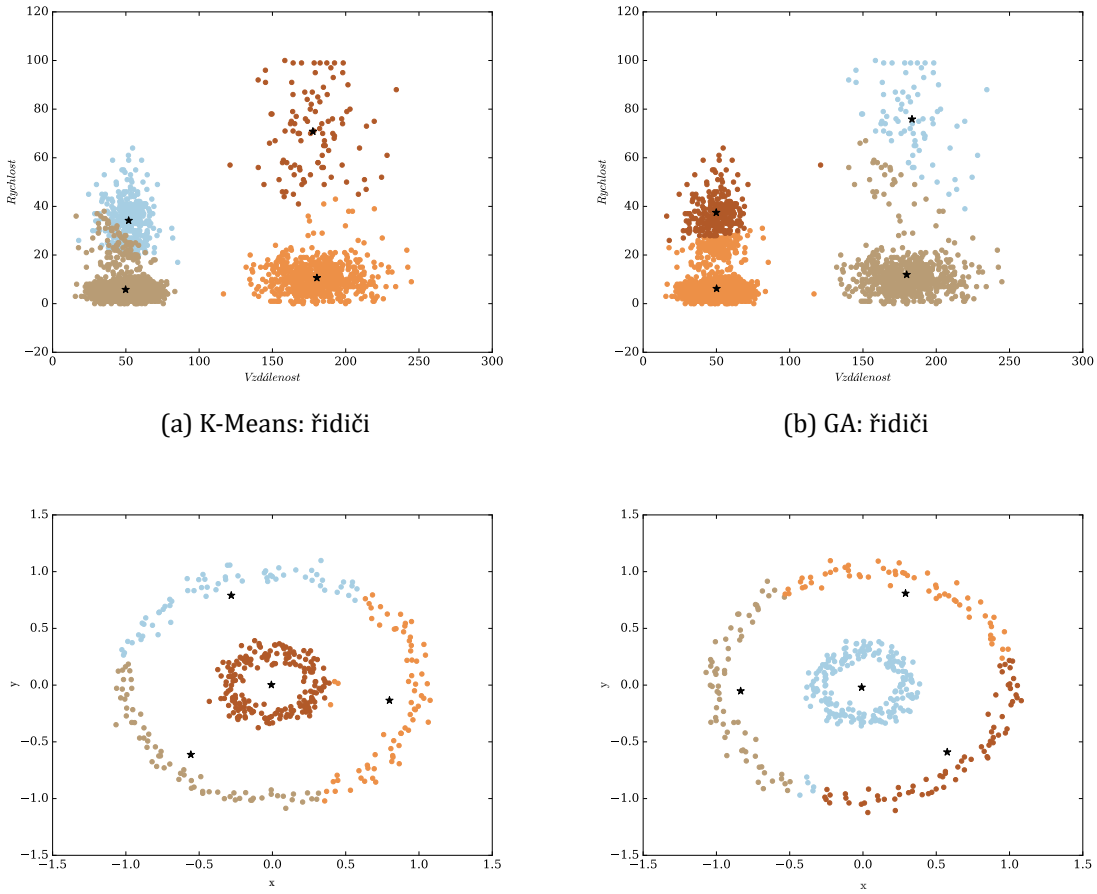
V programovém kódu 8.9 vstupuje populace do metody `_mutation` a jednotliví jedinci (chromozomy) jsou postupně nahrazeny centroidy vrácenými z ohodnocení třídou `GetDaviesBouldin`. Tato metoda navrací novou instanci třídy `Population`.

```
def _mutation(population, fnGetFitness, X):
    fitness = []
    interlen = []
    temp_chromosom = []
    for i, chromosome in enumerate(population.Chromosomes):
        db, ln, centroids = fnGetFitness(chromosome)
        fitness.append(db)
        interlen.append(ln)
        temp_chromosom.append(centroids)
    return Population(temp_chromosom, fitness, interlen)
```

Programový kód 8.9: Mutace populace v GDP

8.2.5 Výsledky implementace genetického algoritmu

V rámci implementace navržených řešení byly provedeny porovnávací testy algoritmu K-průměrů a genetického algoritmu pro uvedené testovací sady *řidiči* a *kruhy*. Na obrázku 8.10 jsou uvedeny grafické výstupy z jednotlivých průběhů algoritmů.



(a) K-Means: řidiči

(b) GA: řidiči

(c) K-Means: kruhy

(d) GA: kruhy

Obr. 8.10: Test implementovaných algoritmů

Centroidy $\mathbf{c}_k$ jsou označeny hvězdičkou. Ostatní barevné hodnoty jsou rozdělená data do shluků S_k. U jednotlivých grafů je možné si všimnout zejména rozdílu v datasetu „kruhy“ a rozdělení jednotlivých centroidů. Více o jednotlivých výstupech vypovídají sledované hodnoty, kterými byly Davies-Bouldinův validační index a průměrná Euklidovská vzdálenost ve shluku dat. Dále byl také sledován potřebný čas pro běh algoritmů. V rámci daného řešení nebylo uvažováno o dynamickém zvýšení či snížení počtu centroidů.

Následují výsledky porovnání obou algoritmů. Každý test byl spuštěn 10× a jsou uvedeny nejlepší výsledky výstupů algoritmů.

```
K-Means řidiči
8 cyklů do konvergence (4m, 16s)
DB Index 0.5167630551103491
Suma průměrných vzdáleností uvnitř shluků: 29.107135396708152

5 cyklů do konvergence (1m, 59s)
DB Index 0.6928435886081372
Suma průměrných vzdáleností uvnitř shluků: 38.22747907985817

2 cykly do konvergence (59s)
DB Index 0.7360602318080822
Suma průměrných vzdáleností uvnitř shluků: 37.830521379605074
```

```
K-Means kruhy
6 cyklů do konvergence (3s, 926ms)
DB Index 0.9987036150188133
Suma průměrných vzdáleností uvnitř shluků: 1.717726567997643

2 cykly do konvergence (1s, 663ms)
DB Index 0.9133526077302941
Suma průměrných vzdáleností uvnitř shluků: 1.6639746612808648

3 cykly do konvergence (2s, 2ms)
DB Index 0.8400359621183359
Suma průměrných vzdáleností uvnitř shluků: 1.4401091736722333
```

```
GA kruhy
2 cykly, Populace = 4 (2s, 72ms)
DB Index 0.77739787348584977
Suma průměrných vzdáleností uvnitř shluků: 1.5172548869635047

2 cykly, Populace = 6 (4s, 49ms)
DB Index 0.97353773307594249
Suma průměrných vzdáleností uvnitř shluků: 1.7154716157662906

2 cykly, Populace = 10 (6s, 2ms)
DB Index 0.8418532846822262
Suma průměrných vzdáleností uvnitř shluků: 1.5331916473395522
```

```
GA kruhy
6 cyklů, populace = 10 (16s, 304ms)
DB Index 0.89134036655984594
Suma průměrných vzdáleností uvnitř shluků: 1.4076740115353803

6 cyklů, populace = 6 (9s, 738ms)
DB Index 0.84332527741694963
Suma průměrných vzdáleností uvnitř shluků: 1.2663053983838712

6 cyklů, populace = 4 (6s, 834ms)
DB Index 0.84007201234023898
Suma průměrných vzdáleností uvnitř shluků: 1.4637060795989525
```

```
GA řidiči
6 cyklů, populace = 4 (1m, 2s)
DB Index 0.46742701779860801
Suma průměrných vzdáleností uvnitř shluků: 27.452260723272207

2 cykly, populace = 4 (23s)
DB Index 0.67284260223514103
Suma průměrných vzdáleností uvnitř shluků: 29.601280430142346
```

Genetický algoritmus ve výše uvedeném testování dosahoval lepších výsledků jak hodnoty Davies-Bouldinova validačního indexu, tak také rozdělení jedinců do jednotlivých shluků v závislosti na jejich Euklidovské vzdálenosti. V případě genetického algoritmu se již výsledek navyšováním počtu opakování ani velikostí populace nezlepšil. Výsledky jsou ovlivněny počátečním náhodným výběrem centroidů, což je vidět například v případě algoritmu K-průměrů v datasetu „řidiči“. Počet opakování i doba trvání je tedy v obou případech zatížena tímto počátečním výběrem.

Výsledky implementace algoritmu pro reálný provoz

V rámci testování výsledků byl použit dataset provozu torrent klientů v kapitole 6.1. Výsledkem testu je schopnost algoritmu porovnat křivky přežití jednotlivých komunikujících uzlů a sdružit je podle podobnosti provozu křivek přežití a míry rizik. Pro kumulativní ohodnocení míry rizik bylo použito v analýze přežití odhadu `Nelson Aalen` z balíčku `lifelines`. V zásadě v tomto případě nejde o zjištění míry některého z rizik, ale o způsob vyjádření chování daného provozu v jiném prostoru.

Uvedený dataset je z databáze importován do supervizoru pomocí balíčku `pandas : DataFrame`. Níže je uvedeno složení vlastního datasetu v `DataFrame`.

```
               IP  a  b     c     d     e  f     g     h     i     j
0    89.176.9.204  0  1  1033  1041  1049  0  0.45  0.78  1.28  2.28
1    192.168.1.54  0  1  1501  1509  1553  0  0.37  0.44  0.51  0.58
2  79.143.185.229  0  1     0     0     0  0  0.00  0.00  0.00  0.00
3    192.168.1.46  0  1  1497  1501  1509  0  0.50  0.75  1.09  1.59
4    192.168.1.66  0  1  1033  1041  1049  0  0.79  0.88  0.98  1.09
```

Tento dataset tvoří genovou matici pro každý SD. Prvních pět hodnot tvoří časovou osu analýzy přežití a následných pět hodnot vlastní hodnoty časové osy. Tento nejjednodušší příklad ověřuje možnosti práce s výstupními hodnotami analýzy přežití a rozlišením jednotlivých křivek přežití.

Pomocí genetického algoritmu jsou zařazeny jednotlivé provozy do shluků. Tyto shluky představují křivky přežití, jejichž rozdíl vlastních hodnot ve vektorech vůči hodnotám centroidu se blíží k nule z pohledu Euklidovské vzdálenosti. Opět byly porovnány výsledky obou algoritmů, jak GA tak K-průměrů.

```
K-Means, 2 cykly do konvergence, (78ms)
DB Index 0.0016132173486533587
Průměrná vzdálenost uvnitř shluku: 0.9918338445433054
Blízká hodnota křivky, shluk 0:
79.143.185.229
Blízká hodnota křivky, shluk 1:
192.168.1.66
89.176.9.204
Blízká hodnota křivky, shluk 2:
192.168.1.54
192.168.1.46
```

Vizuálním ověřením je možné potvrdit správné párování jednotlivých IP adres ve shluku. Shluk číslo 1 obsahuje dvě adresy, z nichž každá představuje komunikaci mezi dvěma počítači s aplikací µTorrent. Další dvě ve shluku číslo 2 zastupují vnitřní IP adresy za směrovačem. Z tohoto pohledu algoritmus K-průměrů přiřadil sobě podobné průběhy do stejného shluku. V dalším kroku byl ověřen genetický algoritmus. Ve výsledném výstupu je zobrazena průměrná vzdálenost uvnitř shluku pro každý cyklus. Jsou zde vidět kroky minimalizace této hodnoty.

```
GA, 4 cykly, populace = 4 (142ms)
Vzdálenost 1. cyklus : [155.33340769869454]
Vzdálenost 2. cyklus : [4.218783667088422]
Vzdálenost 3. cyklus : [2.211733592974675]
Vzdálenost 4. cyklus : [0.9918338445433054]

DB Index 0.0016132173486533587
Průměrná vzdálenost uvnitř shluku: 0.99183384454330537
Blízká hodnota křivky, shluk 0:
79.143.185.229
Blízká hodnota křivky, shluk 1:
89.176.9.204
192.168.1.66
Blízká hodnota křivky, shluk 2:
192.168.1.54
192.168.1.46
```

Z časového pohledu byl výpočet příkladu s reálným provozem pomocí genetického algoritmu náročnější než výpočet pomocí algoritmu K-průměrů. Výsledek profilace algoritmu ukazuje časovou náročnost vytváření nových instancí tříd `Population`. Při opakovaném testování byla objevena chybovost algoritmu, a to konkrétně přiřazení stejné IP adresy do dvou rozdílných shluků. Tato chyba nastávala v případě, že algoritmus ukončil výpočet s výsledkem hodnoty DB indexu $DB > 0,01$.

Navržený GA algoritmus pracuje pouze s centroidy jako individuem a neudržuje data ve shluku, jako je tomu v případě K-průměrů. Při vytvoření populace pomocí volání metody `random.choice()` dochází při malém počtu chromozomu k paradoxu výběru shodných centroidů. Genetický algoritmus není poté schopný danou chybu napravit. Tento paradox je možné ovlivnit zvýšením počtu chromozomů v populaci, případně zvýšit hodnotu mutace. Od počtu deseti chromozomů v populaci již

k chybě nedocházelo. Pokud jsou porovnány výsledky obou algoritmů, při malém počtu vstupních hodnot oba algoritmy dospěly ke konvergenci a stejným hodnotám.

Níže je uveden výstup populace o velikosti čtyř chromozomů při dané chybě. Písmenem *S* jsou označeny shodně zvolené počáteční centroidy.

```
Chromozom 1:
[[[  0.0e+00   1.0e+00   1.0e+03   0.0e+00   7.9e-01   8.8e-01]
  [  0.0e+00   1.0e+00   1.0e+03   0.0e+00   4.5e-01   7.8e-01]  S
  [  0.0e+00   1.0e+00   1.0e+03   0.0e+00   4.5e-01   7.8e-01]] S
Chromozom 2:
 [[  0.0e+00   1.0e+00   1.5e+03   0.0e+00   3.7e-01   4.4e-01]  S
  [  0.0e+00   1.0e+00   1.0e+03   0.0e+00   4.5e-01   7.8e-01]
  [  0.0e+00   1.0e+00   1.5e+03   0.0e+00   3.7e-01   4.4e-01]] S
Chromozom 3:
 [[  0.0e+00   1.0e+00   1.5e+03   0.0e+00   5.0e-01   7.5e-01]
  [  0.0e+00   1.0e+00   1.5e+03   0.0e+00   3.7e-01   4.4e-01]
  [  0.0e+00   1.0e+00   1.0e+03   0.0e+00   4.5e-01   7.8e-01]]
Chromozom 4:
 [[  0.0e+00   1.0e+00   0.0e+00   0.0e+00   0.0e+00   0.0e+00]  S
  [  0.0e+00   1.0e+00   0.0e+00   0.0e+00   0.0e+00   0.0e+00]  S
  [  0.0e+00   1.0e+00   1.0e+03   0.0e+00   4.5e-01   7.8e-01]]]
```

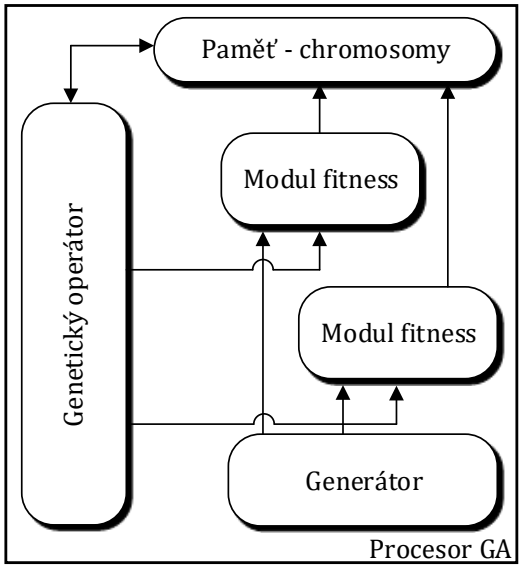
8.3 Implementace genetického algoritmu v FPGA

Pro možné budoucí využití navrženého algoritmu v navazujících projektech byl v průběhu řešení dizertační práce rozpracován návrh implementace genetického algoritmu do programovatelné síťové karty osazené hradlovým polem FPGA. Tato část modulu by mohla sloužit k detekci provozu v pasivních optických sítích xPON (*Passive Optical Network*) mezi OLT (*Optical Line Terminal*) a ONU (*Optical Network Unit*). S využitím jazyka VHDL byl vytvořen část firmware v programu ISE firmy XILINX.

Procesor genetického algoritmu je složen z jednotlivých dílčích modulů, jako je modul genetického operátoru `GO`, modul `fitness`, modul pseudonáhodného generátoru `ASGG` (*Alternating Stop and Go Generator*) a modul paměti, viz obrázek 8.11.

Modul `GO` představuje jádro genetického algoritmu, kde jsou zastoupeny jeho jednotlivé funkce, jako je proces křížení, mutace a selekce. Zjednodušené schéma je uvedeno v příloze C. Modul `fitness` zajišťuje definování funkce ohodnocení jedince. Modul generátoru `ASGG` zajišťuje inicializaci náhodné populace.

V genetickém algoritmu je velkou měrou využito uvedených pseudonáhodných generátorů. Proto byl jako první detailněji rozpracován jejich návrh, viz obrázek 8.12. Prozatím jako jediný modul je celkově hotový a byl testován. Detaily testování jsou uvedeny v příloze vlastní tvorby [A5]. Je nutné provést úpravu možnosti nastavení generování šířky populace, binárního slova za běhu systému, a také provést konečné řešení zapojení jednotlivých portů.

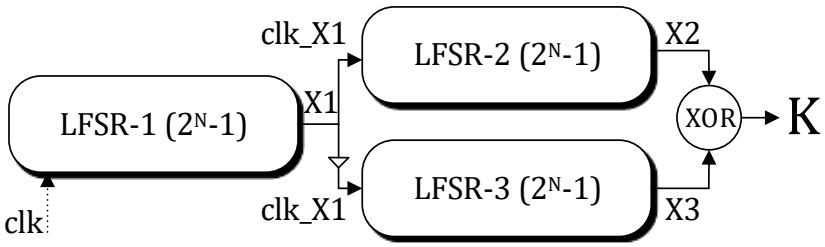


Obr. 8.11: FPGA procesor genetického algoritmu

Je složen ze tří LFSR (*Linear Feedback Shift Register*) registrů a cyklicky generuje pseudonáhodnou sekvenci stavů. Jeho časová komplexnost T_A odpovídá jeho složení z jednotlivých LFSR registrů a je rovna (8.8).

$$T_A = (rX2 \,\overline{\vee}\, rX3) = 1 + t(rX2) + t(rX3) \tag{8.8}$$

Nově zde bylo provedeno statistické testování implementace generátorů ve VHDL pomocí NIST STS [84], (*A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications*). Výsledky byly taktéž publikovány ve zmíněném příspěvku.



Obr. 8.12: ASGG generátor

9 ZÁVĚR A DISKUZE DOSAŽENÝCH VÝSLEDKŮ

V rámci dizertační práce byl navržen a implementován nový prvek – sonda síťových anomálií. Dále byly vypracovány vlastní algoritmy pro behaviorální analýzu, které využívají genetické algoritmy. Cíle definované v příslušné kapitole se podařilo během řešení naplnit.

Samotná definice „anomálie“ představuje velmi široký pojem a je těžké ji obecně definovat. Například při útoku typu DoS se může jednat o dočasný zvýšený zájem o určitou službu, která vyvolá falešně pozitivní reakci. Dané řešení se zabývalo možností detekovat takové cykly provozu, které jsou generovány v jiných časových úsecích a v rozdílných sítích. Příkladem je možnost detekce komunikace ransomware nebo serverů botnet sítě, které periodicky generují provoz. V tomto případě je možné se zaměřit na životní cyklus, a jako podklad pro rozhodovací proces získat přehled komunikujících uzlů s podobným průběhem provozu. Ve výsledku je již možné se zaměřit pouze na vybraná spojení a určit, zda se jedná o nevyžádaný provoz.

Vlastní implementace algoritmů a navržený model sondy síťových anomálií pak umožňují převod zachyceného datového provozu do křivek přežití a je provedena analýza datových toků na základě jejich podobnosti. Hodnoty představující křivky přežití jsou zpracovány genetickým algoritmem, kde je vyhodnoceno jejich seskupení a souvztažnost.

Prvním vytyčeným cílem bylo analyzovat současný stav prostředků a používaných algoritmů k detekci provozu. V kapitolách 1.2 až 1.3.5 byl probrán teoretický podklad a jejich současné využití. Následně v kapitole 1.4 bylo provedeno praktické měření analyzátorem ENDACE PROBE EP7010-PS-FC a byl detailněji zkoumán princip fungování protokolu NetFlow. Byla naprogramována vlastní sonda založená na měření RTT sbírající data ze sondy RIPE NNC. Testován byl také zapůjčený IDS systém GAiA společnosti Check Point Software Technologies Ltd.

V rámci analýzy současného stavu byla naprogramována první část modelu, a to kolektor NetFlow nazvaný GDP. Byl použit jazyk Python ve verzi 3. Správná funkčnost byla testována v zapojení se směrovači v programu GNS3. Dále byly použity laboratorní přepínače k zasílání NetFlow verze 1 a 5. Byl proveden sběr dat z veřejné a laboratorní sítě.

Druhým vytyčeným cílem bylo navrhnout novou metodu detekce anomálií provozu. Kapitola 4 upřesňuje a detailněji specifikuje daný cíl. Byl proveden úvodní srovnávací test programu OMNeT++ a Python. Původně zamýšlené využití programu pro simulaci sítí OMNeT++ se ukázalo být velmi složité z pohledu programování a časové náročnosti. Oproti tomu Python poskytuje dostačující podporu a potřebné knihovny pro výzkum a vývoj.

Byl proveden test na základě hypotézy využití analýzy přežití pro vlastní roz-

lišení provozu. V tomto testování bylo využito torrent klientů, vlastního botnet serveru, a také datasetu provozu z ČVUT. Z výsledků bylo usouzeno, že je teoreticky možné vytvořit vzory provozu na základě křivek analýzy přežití a sledovat jejich životní cyklus.

Další stanovené cíle se již vzájemně doplňují. V rámci naplnění cíle vývoje algoritmu či skupiny algoritmů vycházejících z algoritmů evolučních bylo provedeno praktické prozkoumání funkčnosti a vlivu operátorů genetických algoritmů, uvedeno v kapitole 7. Byl zde navržen a naprogramován vlastní genetický algoritmus v jazyce Python.

Genetické algoritmy mohou najít mnohé uplatnění. Velmi důležité je zvolení koncepce ověřování výsledků – ohodnocující funkce. Ta vždy musí odpovídat danému problému. Dále je také vhodné určit, zda má smysl použít genetický algoritmus či nikoli. Genetické algoritmy patří mezi heuristické metody řešení daného problému. Jsou vhodné zejména tam, kde není možné jinou metodou či algoritmem v rozumném čase dospět k výsledku řešení nebo není známá funkce daného řešení. Jedná se zejména o NP–těžké úlohy. V mnoha úlohách mohou genetické algoritmy selhávat a nepředstavují univerzální nástroj.

Kapitola 8 již představuje plnění cíle vytvoření vlastního modelu. V rámci plnění tohoto cíle byl dokončen návrh druhého modulu síťové sondy – supervizor. Byl zde implementován genetický algoritmus, který je určen pro rozdělení zachyceného provozu převedeného do křivek přežití do jednotlivých shluků na základě Euklidovské vzdálenosti. Jedná se tedy o porovnání bodů všech průběhů, vektorů jednotlivých spojení. Původně uvažované řešení zjištění vzdáleností křivek za pomocí minimalizace kostry grafu síťových spojení se ukázalo být neefektivní v tom směru, že by samotný výstup obsahoval pouze jeden shluk nejkratších vzdáleností.

Navržený a implementovaný genetický algoritmus částečně využívá principu algoritmu K-průměrů, ale pracuje pouze se samotnými centroidy, které představují chromozom řešení. Následně aktualizuje shluky a centroidy na základě Euklidovské vzdálenosti. V paměti nejsou udržovány jednotlivé shluky. Algoritmus minimalizuje hodnotu Davies-Bouldinova validačního indexu a hodnotu sumy Euklidovských vzdáleností všech shluků.

Samotné řešení poskytuje práci s pevně stanoveným počtem shluků. Uvedený algoritmus je vhodné rozšířit o možnost dynamicky zvyšovat či snižovat počet těchto shluků a o možnost nastavit rozpětí Euklidovské vzdálenosti. V dalším možném vylepšení je vhodné provést paralelizace genetického algoritmu a pracovat na efektivnosti programového kódu. Samotný genetický algoritmus je připraven na práci s vektory libovolné délky. Není tedy omezen na zpracování pouze výsledků analýzy přežití. Tato představovala jedno z možných vyjádření vzorů provozu. Provedenými testy se potvrdilo, že pomocí navrženého algoritmu lze ve zvoleném časovém úseku rozlišit

a párovat datové provozy podle jejich průběhu.

Na provedený výzkum by mohla navázat implementace algoritmů do programovatelných hradlových polí FPGA síťových karet. Projekty zaměřující se na implementaci FPGA dále navazují na vlastní řešené téma. Jedná se o jeden ze tří autorem vypsaných projektů, a to o možnost detekce dat v xPON sítích mezi koncovými a řídícími jednotkami. Během řešení dizertační práce byl rozpracován návrh nasazení genetického algoritmu a proveden návrh a testování pseudonáhodných generátorů. Generátory jako takové hrají důležitou roli a jsou genetickými algoritmy hojně využívány. Konkrétně se jednalo o návrh ASGG a GEFFE pseudonáhodného generátoru.

V oblasti detekce anomálií by bylo zajímavé zaměřit další výzkum na metody umělého imunitního systému, které v současné době představují vhodného nástupce neuronových sítí a genetických algoritmů a kombinují jejich možnosti. Zajímané využití v konvergovaných sítích a jednotlivých aplikacích by mohly nalézt soutěživé algoritmy.

Dílčí výsledky byly průběžně publikovány a všechny vytyčené cíle dizertační práce byly dosaženy.

LITERATURA

[1] Cisco Devnet: APIs, SDKs, Sandbox, and Community for Cisco Developers, *Cisco* [online]. 2015 [cit. 2016-11-04]. Dostupné z URL: <https://developer.cisco.com/site/devnet/home/index.gsp>.

[2] STALLINGS, William. Software-Defined Networks and OpenFlow: The Internet Protocol Journal, Volume 16, No. 1. CISCO INC. *Cisco* [online]. [cit. 2015-04-02]. Dostupné z URL: <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html#reference1>.

[3] *Cisco* [online]. 2015 [cit. 2015-04-04]. Dostupné z URL: <http://www.cisco.com/c/en/us/index.html>.

[4] ZAKRZEWSKA, A, S RUEPP a M.S. BERGER. *Towards Converged 5G Networks: Challenges and Current Trends*. Saint Petersburg: ITU Kaleidoscope, 2014. Dostupné z URL: <http://www.itu.int/en/ITU-T/academia/kaleidoscope/2014/Pages/default.aspx>.

[5] Prolexic Quarterly Global DDoS Attack Report Q1 2014. In: Prolexic Technologies., Inc., 2014.

[6] Global Application & Network Security Report 2014-2015. In: *GLOBAL APPLICATION & NETWORK SECURITY REPORT 2014-2015* [online]. 2015 [cit. 2015-04-04]. Dostupné z URL: <http://www.radware.com>.

[7] Q42014 State of the Internet – Security Report. *State of the Internet: brought to you by Akamai* [online]. 2015 [cit. 2015-03-31]. Dostupné z URL: <http://www.stateoftheinternet.com/resources-web-security-2014-q4-internet-security-report.html>.

[8] DDoS attack that disrupted internet was largest of its kind in history, experts say. *Theguardian* [online]. London: Guardian News and Media Limited, 2017 [cit. 2017-06-04]. Dostupné z URL: <https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>.

[9] CSIRT.CZ, *Upozornění pro ISP na možnou hrozbu útoku na zákazníky xDSL* CSIRT.CZ, [cit. 201 5-01-29]. [online]. Dostupné z URL: <https://www.csirt.cz/page/2422/.>.

[10] CVEdetails.com the ultimate security vulnerability data source. MITRE CORPORATION. [online]. 2015. vyd. [cit. 2014-11-12]. Dostupné z URL: <http://www.cvedetails.com/>.

[11] BRODKIN, Jon. Netflix packets being dropped every day because Verizon wants more money. [online]. 2014 [cit. 2014-10-29]. Dostupné z URL: <http://arstechnica.com/....-wants-more-money/>.

[12] VARDI, Y. Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data. *Journal of the American Statistical Association*. 1996, vol. 91, v. 433. DOI: 10.2307/2291416.

[13] Catalyst 6500 Release 12.2SX Software Configuration Guide - SPAN, RSPAN, and ERSPAN. CISCO SYSTEMS, Inc. *Cisco* [online]. 2006 [cit. 2015-04-18]. Dostupné z URL: <http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/span.html>.

[14] MEMON, Rashida A., Sameer QAZI a Adnan A. FAROOQUI. Network tomography using genetic algorithms. *TENCON 2012 IEEE Region 10 Conference* [online]. IEEE, 2012, s. 1-6 [cit. 2015-01-14]. DOI: 10.1109/TENCON.2012.6412313. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6412313>.

[15] RIPE NCC. *RIPE Network Coordination Centre: RIPE Labs* [online]. 2015 [cit. 2015-01-26]. Dostupné z URL: <https://labs.ripe.net/>.

[16] Algoritmy.net, Jan Neckář, 2016 [online]. [cit. 2017-04-19]. Dostupné z: <https://www.algoritmy.net/>.

[17] FERNANDEZ, Eduardo B. *Security patterns in practice: designing secure architectures using software patterns*. United Kingdom: John Wiley & Sons, Ltd., 2013, xxi, s. 558. Wiley series in software design patterns. ISBN 978-1-119-99894-5.

[18] SUN, Daniel, Min FU, Liming ZHU, Guoqiang LI a Qinghua LU. Non-Intrusive Anomaly Detection With Streaming Performance Metrics and Logs for DevOps in Public Clouds: A Case Study in AWS. *IEEE Transactions on Emerging Topics in Computing* [online]. 2016, **4**(2), 278-289 [cit. 2017-04-10]. DOI: 10.1109/TETC.2016.2520883. ISSN 2168-6750. Dostupné z: <http://ieeexplore.ieee.org/document/7389388/>.

[19] BEREZIŃSKI, Przemysław, Bartosz JASIUL a Marcin SZPYRKA. An Entropy-Based Network Anomaly Detection Method. *Entropy* [online]. 2015, **17**(4), 2367-2408 [cit. 2017-04-08]. DOI: 10.3390/e17042367. ISSN 1099-4300. Dostupné z: <http://www.mdpi.com/1099-4300/17/4/2367/>.

[20] Rényi Entropy. *Wolfram MathWorld: The Web's Most Extensive Mathematics Resource* [online]. 2017 [cit. 2017-04-08]. Dostupné z: <http://mathworld.wolfram.com/RenyiEntropy.html>.

[21] JAVED, Mobin, Ayesha Binte ASHFAQ, M. Zubair SHAFIQ a Syed Ali KHAYAM. *On the Inefficient Use of Entropy for Anomaly Detection* [online]. 2009, , 369 [cit. 2017-04-08]. DOI: 10.1007/978-3-642-04342-0_28. Dostupné z: <http://link.springer.com/10.1007/978-3-642-04342-0_28>.

[22] SMITH, Lindsay I. *A tutorial on Principal Components Analysis* [online]. 2002, [cit. 2017-04-10]. Dostupné z: <http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf>.

[23] HUANG, Hong, Hussein AL-AZZAWI a Hajar BRANI. Network Traffic Anomaly Detection. *Semantic Scholar* [online]. Allen Institute for Artificial Intelligence, 2014, , 26 [cit. 2017-04-11]. arXiv:1402.0856v1 [cs.CR]. Dostupné z: <https://pdfs.semanticscholar.org/2ad0/8da69a014691ae76cf7f53534b40b412c0e4.pdf>.

[24] NSR – Network Security Research. 2014–2016, [cit. 2017-04-10]. Dostupné z: <http://nsr.utko.feec.vutbr.cz/probev3.php>.

[25] Principal Component Analysis in 3 Simple Steps. *Plotly* [online]. 2015 [cit. 2017-04-11]. Dostupné z: <https://plot.ly/ipython-notebooks/principal-component-analysis/>.

[26] *Matematická biologie* [online]. Brno: Institut biostatistiky a analýz Masarykovy univerzity, 2016 [cit. 2017-05-11]. Dostupné z: <http://portal.matematickabiologie.cz>.

[27] L.N. De Castro a J. Timmis: Artificial Immune Systems: a new computational intelligence approach, Springer, 1st Edition., XVIII, 380 s., 2002, ISBN 978-1-85233-594-6

[28] DIPANKER Dasgupta: Advances in Artificial Immune Systems, *IEEE Computational Intelligence Magazine*, 2006, Vydání. 1, č. 4, s. 40-49, 2006, doi:10.1109/MCI.2006.329705 Dostupné z: <http://ieeexplore.ieee.org/document/4129847/>.

[29] U. AICKELIN a D. Dasgupta: Artificial Immune Systems, kapitola v knize, Search Methodologies:Introductory Tutorials in optimization and decision support techniques, 2003, Springer, s. 375-399, ISBN 978-1-4614-6939-1. Dostupné z: <http://eprints.nottingham.ac.uk/3345/1/aickelin2014.pdf>.

[30] Artificial Immune Systems. *Dr. Mark Read* [online]. Sydney: The University of Sydney, s. 25 [cit. 2017-04-09]. Dostupné z: <http://markread.info/pubs/ais.pdf>.

[31] SOMVANSHI, Divya a R.D.S. YADAVA. Boosting Principal Component Analysis by Genetic Algorithm. *Defence Science Journal* [online]. 2010, **4**(60), 7 [cit. 2017-04-12]. DOI: 10.1.1.902.7675. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.902.7675&rep=rep1&type=pdf>.

[32] RIVERA-GALLEGO, Wilson. A GENETIC ALGORITHM FOR SOLVING THE EUCLIDEAN DISTANCE MATRICES COMPLETION PROBLEM. *SAC* [online]. 1998, , 5 [cit. 2017-04-12]. ACM l-S81 13-086-4199/000. Dostupné z: <http://slapper.apam.columbia.edu/bib/papers/river_b_99.pdff>.

[33] ZHANG, Y., Ge, Z., Greenberg, A. a Roughan, M. Network anomography. *(2005) Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC,* s. 317-330. [cit. 2017-04-11] <http://conferences.sigcomm.org/imc/2005/papers/imc05efiles/zhang/zhang.pdf>.

[34] SILVERAY, F.; Diot, C.; Taft, N.; Govindan, R.: ASTUTE: Detecting a Different Class of Traffic Anomalies. In SIGCOMM Proceedings, New Delhi, India, Září 2010 [cit. 2017-04-03], [online]. DOI: http://doi.acm.org/10.1145/1851275.1851215. Dostupné z: <http://www.sigcomm.org/ccr/papers/2010/October/1851275.1851215>.

[35] *Stratosphere IPS Project* [online]. 2015 [cit. 2017-04-12]. Dostupné z: <https://stratosphereips.org/>.

[36] CHANDRASEKARAN, Balakrishnan. *Survey of Network Traffic Models* [online]. , 8 [cit. 2017-04-12]. Dostupné z: <http://www.cse.wustl.edu/~jain/cse567-06/ftp/traffic_models3.pdf>.

[37] *PRTG Network Monitor - Powerful Network Monitoring Software* [online]. Nuremberg: Paessler, 2017 [cit. 2017-04-04]. Dostupné z: <https://www.paessler.com>.

[38] Targeted Attacks Against Corporate Inboxes - a Gmail Perspective RSA 2017. *SlideShare* [online]. 2017 [cit. 2017-02-18]. Dostupné z: </www.slideshare.net/elie-bursztein/>.

[39] TensorFlow: An open–source software library for Machine Intelligence. *TensorFlow* [online]. [cit. 2017-02-18]. Dostupné z: <https://www.tensorflow.org>.

[40] Software Defined Specification Environment for Networking. *XILINX: all programmable* [online]. 2015 [cit. 2015-01-03]. Dostupné z URL: <http://www.xilinx.com/applications/wired-communications/sdnet.html>.

[41] NIFIC Handbook: The Liberouter Project Team. CESNET, z.s.p.o. *LIBEROUTER* [online]. 2009 [cit. 2014-10-04]. Dostupné z URL: <http://www.liberouter.org/package_releases/nific-3.2.0/handbook/nific-handbook.html#chap-config-local-vif>.

[42] SDM. *Liberouter / Cesnet TMC group | Programmable hardware | Liberouter / Cesnet TMC group* [online]. 2015 [cit. 2015-04-03]. Dostupné z URL: <https://www.liberouter.org/technologies/sdm/>.

[43] NetCOPE Development Framework - INVEA-TECH. *INVEA-TECH - High-Speed Networking and FPGA Solutions* [online]. INVEA-TECH a.s., Copyright © 2007 — 2015 [cit. 2015-01-21]. Dostupné z URL: <https://www.invea.com/en/products-and-services/fpga-development-kit/netcope>.

[44] PAPADONIKOLAKIS, Markos, Christos-Savvas BOUGANIS, George CONSTANTINIDES, Dominique LAVENIER, Van-Hoa NGUYEN, Marcin PIETRON, Maciej WIELGOSZ, Dominik ZUREK, Ernest JAMRO a Kazimierz WIATR. Performance comparison of GPU and FPGA architectures for the SVM training problem. *2009 International Conference on Field-Programmable Technology* [online]. 2009, s. 292-302 [cit. 2015-04-18]. DOI: 10.1007/978-3-642-36424-2_25.

[45] Introduction to Cisco IOS NetFlow - A Technical Overview. CISCO SYSTEMS, INC. *CISCO* [online]. 2012 [cit. 2016-01-22]. Dostupné z: </http://www.cisco.com/NetFlow/>.

[46] Plixer: Flow Analytics. PLIXER INTERNATIONAL, INC. *Plixer - Malware Incident Response* [online]. 2016 [cit. 2016-01-20]. Dostupné z: <https://www.plixer.com/Scrutinizer-Netflow-Sflow/flow-analytics.html>.

[47] MCDONNELL, John R, Robert G REYNOLDS a David B FOGEL. *Evolutionary programming IV: proceedings of the Fourth Annual Conference on Evolutionary Programming*. Cambridge, Mass.: MIT Press, c1995, xx, 805 p. ISBN 02-621-3317-2.

[48] HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.

[49] STORN, Rainer a Kenneth Price,. *Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces* In: Journal

of Global Optimization 11: 341–359, 1997, Kluwer Academic Publishers. Dostupné z URL: <http://jaguar.biologie.hu-berlin.de/~wolfram/pages/seminar_theoretische_biologie_2007/literatur/schaber/Storn1997JGlobOpt11.pdf>

[50] POLI, Riccardo, W LANGDON a Nicholas F MCPHEE. *A field guide to genetic programming* [online]. [S.l.: Lulu Press], 2008, xiv, 233 s. [cit. 2015-01-25]. ISBN 978-1-4092-0073-4. Dostupné z URL: <http://www.gp-field-guide.org.uk/>.

[51] DEB, Kalyanmoy, Amrit PRATAP, Sameer AGARWAL a T MEYARIVAN. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* [online]. 2012, **6**(2), 16 [cit. 2017-06-06]. ISSN 1089-778X. 1089-778X(02)04101-2. Dostupné z URL: <https://www.iitk.ac.in/kangal/Deb_NSGA-II.pdf>.

[52] HINTERDING, R. Gaussian mutation and self-adaption for numeric genetic algorithms. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation* [online]. IEEE, 1995, , 384- [cit. 2017-04-13]. DOI: 10.1109/ICEC.1995.489178. ISBN 0-7803-2759-4. Dostupné z: <http://ieeexplore.ieee.org/document/489178/>.

[53] MAN, Kim F., Kit S. TANG, Sam KWONG a Wolfgang A. HALANG. *Genetic algorithms for control and signal processing*. 1. S.l.: Springer, 1997. ISBN 978-144-7112-419.

[54] ALBA, Enrique. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters* [online]. 2002, vol. 82, issue 1, s. 7-13 [cit. 2015-04-18]. DOI: 10.1016/s0020-0190(01)00281-2. Dostupné z URL: <http://atarazanas.sci.uma.es/docs/tesisuma/16640299.pdf>.

[55] SCOTT, Stephen D., Ashok SAMAL a Shared SETH. HGA. *Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays - FPGA '95* [online]. New York, New York, USA: ACM Press, 1995, s. 53-59 [cit. 2015-04-25]. DOI: 10.1145/201310.201319. Dostupné z URL: <http://portal.acm.org/citation.cfm?doid=201310.201319>.

[56] FERNANDO, P.R., S. KATKOORI, D. KEYMEULEN, R. ZEBULUM a A. STOICA. Customizable FPGA IP Core Implementation of a General-Purpose Genetic Algorithm Engine. *IEEE Transactions on Evolutionary Computation* [online]. 2010, vol. 14, issue 1, s. 133-149 [cit. 2015-04-25]. DOI:

10.1109/TEVC.2009.2025032. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5299091>.

[57] UNIVERSITY OF CALIFORNIA'S SAN DIEGO SUPERCOMPUTER CENTER. *CAIDA: Center for Applied Internet Data Analysis* [online]. 2015 [cit. 2015-01-18]. Dostupné z URL: <http://www.caida.org/home/>.

[58] NORMAN, Geoffrey R. a David L. Streiner. Biostatistics: the bare essentials. 3rd ed. Shelton, Conn.: People's Medical Pub. House, 2008. ISBN 978-155-0093-476

[59] S.C.S. SILVA, R.M.P. Silva, R.C.G. Pinto, a R.M. Salles, "Botnets: A survey,"Computer Networks, vol. 57, s. 378–403, 2013.

[60] GCAT: A fully featured backdoor that uses Gmail as a C&C server, GitHub.

[61] Threat Spotlight: Dyre/Dyreza: An Analysis to Discover the DGA. *Cisco Blogs* [online]. USA, San Jose: Cisco Systems, 2015 [cit. 2017-06-05]. Dostupné z URL: <http://blogs.cisco.com/security/talos/threat-spotlight-dyre>.

[62] OpenSim Ltd. *OMNeT++ Discrete Event Simulator* [online]. vydání © 2001-2015. [cit. 2015-02-27]. Dostupné z URL: <http://omnetpp.org/>.

[63] *ReliaSoft Corporation* [online]. vydání 2015. [cit. 2015-02-28]. Dostupné z URL: <http://reliasoft.com/>.

[64] MELAKESSOU, Foued. NETWORK ANALYSIS AND ROUTING EVALUATION: THE NARVA MODULE. In: [online]. SCILABTEC 6th International Users Conference: University of Luxembourg, 2014 [cit. 2015-04-28]. Dostupné z URL: <http://orbilu.uni.lu/bitstream/10993/20384/1/UniversityLuxembourg_ScilabTEC2014.pdf>.

[65] GitHub, Fabric: Simple, Pythonic remote execution and deployment, GitHub, [online] 2016 [cit. 2017-04-16]. Dostupné z: <https://github.com/fabric/fabric>.

[66] A simple botnet written in Python. *Charlesleifer* [online]. [cit. 2017-04-19]. Dostupné z: <http://charlesleifer.com/blog/simple-botnet-written-python/>.

[67] Stratosphere IPS, Dataset, 2015. [online]. [cit. 2017-04-19]. Dostupné z: <https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-1/>.

[68] *GPLAB: A Genetic Programming Toolbox for MATLAB* [online]. 2014 [cit. 2015-04-28]. Dostupné z URL: <http://gplab.sourceforge.net/index.html>.

[69] RANGAIAH, Gade Pandu. *Multi-objective optimization: techniques and applications in chemical engineering / editor Gade Pandu Rangaiah*. Hackensack, N.J.: World Scientific, c2009, xvii, 435 p. ISBN 98-128-3651-9.

[70] ECKART ZITZLER, Lothar Thiele. *Evolutionary Multi-Criterion Optimization First International Conference, EMO 2001 Zurich, Switzerland, March 7-9, 2001 Proceedings*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2001. ISBN 978-354-0447-191.

[71] NGPM – A NSGA-II Program in Matlab v1.4. *MATLAB Central* [online]. 2011 [cit. 2015-04-28]. Dostupné z URL: <http://www.mathworks.com/matlabcentral/fileexchange/31166-ngpm-a-nsga-ii-program-in-matlab-v1-4>.

[72] VAN VELDHUIZEN, D.A. a G.B. LAMONT. On measuring multiobjective evolutionary algorithm performance. In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*. IEEE, 2000, s. 204-211. ISBN 0-7803-6375-2. DOI: 10.1109/CEC.2000.870296. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=870296>.

[73] Benchmarks. *DEAP documentation* [online]. 2017 [cit. 2017-04-16]. Dostupné z: <http://deap.readthedocs.io/en/master/api/benchmarks.html>.

[74] HÉGR, Tomáš a Leoš BOHÁČ. Impact of Nodal Centrality Measures to Robustness in Software-Defined Networking. *ADVANCES IN ELECTRICAL AND ELECTRONIC ENGINEERING* [online]. 2014, roč. 2014, č. 12, s. 252-259 [cit. 2015-04-03]. Dostupné z URL: <http://advances.utc.sk/index.php/AEEE/article/download/1208/982>.

[75] VEGA-RODRÍGUEZ, Miguel A. et al Genetic Algorithms Using Parallelism and FPGAs: The TSP as Case Study. [online]. IEEE Computer Society, 2005, s. 7 [cit. 2015-04-05]. Dostupné z URL: <http://users.dsic.upv.es/~rgutierrez/download/vgasg05.pdf>.

[76] SKORPIL, Vladislav a Stanislav KAMBA. Back propagation and Genetic Algorithms for control of the network element. *2011 34th International Conference on Telecommunications and Signal Processing (TSP)*. 2011. DOI: 10.1109/tsp.2011.6043735.

[77] Converged Systems. *SIX research centre: Sensor, Information and Communication Systems* [online]. 2015 [cit. 2015-04-03]. Dostupné z URL: <http://www.six.feec.vutbr.cz/programs/converged-systems/>.

[78] GEN, Mitsuo, Runwei CHENG a Lin LIN. *Network models and optimization: multiobjective genetic algorithm approach*. Londýn: Springer, c2008, xiv, 692 p. ISBN 18-480-0180-0.

[79] Network Engineering & Security Group. *First release of the NETA framework.* [online]. 2015 [cit. 2015-04-26]. Dostupné z URL: <http://nesg.ugr.es/index.php/en/downloads/viewcategory/8-v100>.

[80] KOTENKO, Igor a Alexander ULANOV. Simulation of Internet DDoS Attacks and Defense. In: KATSIKAS, Sokratis K. *Information security: 9th international conference, ISC 2006, Samos Island, Greece, August 30-September 2, 2006 : proceedings*. New York: Springer, c2006, s. 327. ISBN 3540383417. DOI: 10.1007/11836810_24. Dostupné z URL: <http://link.springer.com/10.1007/11836810_24>.

[81] NetFlow Version 5 - Plixer.com. *PLIXER* [online]. Kennebunk: Plixer International, 2017 [cit. 2017-04-05]. Dostupné z: <https://www.plixer.com/support/netflow_v5.html>.

[82] *An introduction to Generative Adversarial Networks (with code in TensorFlow)* [online]. Aylien, 2016 [cit. 2017-04-12]. Dostupné z: <http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow>.

[83] *Clustering*, scikit learn [online]. 2016 [cit. 2017-05-11]. Dostupné z: <http://scikit-learn.org/stable/modules/clustering.html>.

[84] NIST.gov - Computer Security Division - Computer Security Resource Center: Cryptographic Toolkit. National Institute of Standards and Technology [online]. 2012 [cit. 2013-11-11]. Dostupné z: <http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html>.

[85] "Introduction to K-means Clustering", DATASCIENCE, 2017. [cit. 2017-06-12], [online]. Dostupné z: <https://www.datascience.com/blog/.

Jednotlivá uváděná RFC

[L1] RFC2460. *Internet Protocol, Version 6 (IPv6): Specification*. Internet Society (ISOC), 1998. Dostupné z URL: <https://tools.ietf.org/html/rfc2460>.

[L2] RFC1122: Requirements for Internet Hosts – Communication Layers [online]. Internet Engineering Task Force, 1989 [cit. 2017-04-03]. Dostupné z: <https://tools.ietf.org/html/rfc1122>.

[L3] RFC792: INTERNET CONTROL MESSAGE PROTOCOL [online]. Internet Engineering Task Force, 1981 [cit. 2017-04-03]. Dostupné z: <https://tools.ietf.org/html/rfc792>.

[L4] RFC7011: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information [online]. Internet Engineering Task Force, 2013 [cit. 2017-04-03]. Dostupné z: <https://tools.ietf.org/html/rfc7011>.

[L5] SNMP Research International, Inc. – SNMP RFCs [online]. [cit. 2017-04-03]. Dostupné z: <http://www.snmp.com/protocol/snmp_rfcs.shtml>.

SEZNAM POUŽITÝCH ZKRATEK

5G	5th Generation Mobile Networks
ACI	Application Centric Infrastructure
ACK	Acknowledge
AIS	Artificial Immune Systems
API	Application Programming Interface
ASGG	Alternating Stop and Go Generator
ASIC	Application Specific Integrated Circuit
CAIDA	Center for Applied Internet Data AnalysisC&CCommand and Control
CSIRT	Computer Security Incident Response Team
CVE	Common Vulnerabilities and Exposures
DGA	Domain Generation Algorithm
DR	Data Recognition
DoS	Denial of Service
DDoS	Distributed Denial of Service
DEAP	Distributed Evolutionary Algorithms in Python
D2D	Direct Device to Device Communication
DBI	Davies-Bouldin Index
DNS	Domain Name System
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
ERSPAN	Encaptulated Remote Switched Port Analyzer
FPGA	Field Programmable Gate Array
GA	Genetic Algorithm
GDP	Genetic Decision Probe

GO	Genetic Operator
HV	Hypervolume
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IGD	Inverse Generational Distance
IOS	Cisco Internetwork Operating System
IoT	Internet of Things
IP	Internet Protocol
IPP	Interrupted Poisson Process
IPv6	Internet Protocol Version 6
IRC	Internet Relay Chat
ISP	Internet Service Provider
KP	Knapsack Problem
LFSR	Linear Feedback Shift Register
LTE	Long Term Evolution
M2MC	Machine to Machine Communication
MV ČR	Ministerstvo vnitra České republiky
MOO	Multi-Objective Optimization
NESG	Network Engineering and & Security Group
NFV	Network Functions Virtualization
NGN-GSI	Next Generation Networks Global Standards Initiative
NIX	Neutral Internet eXchange
NFL	No Free Lunch Theorem

NP	Nondeterministic Polynomial
NSGA	Niched Sharing Genetic Algorithm
NT	Network Tomography
NTPv2	Network Time Protocol version 2
OLT	Optical Line Terminal
ONU	Optical Network Unit
OS	Operating System
PCA	Principal Components Analysis
PING	Packet InterNet Gropér
RAM	Random-access Memory
RFC	Request for Comments
RIPE NNC	The Réseaux IP Européens Network Coordination Centre
RSPAN	Remote Switched Port Analyzer
RTT	Round–Trip Time
SDN	Software Defined Network
SMA	Simple Moving Average
SNMP	Simple Network Management Protocol
SPAN	Switched Port Analyzer
SIX	Sensor, Information and Communication Systems
SSH	Secure Shell
SVD	Singular Value Decomposition
TAP	Test Access Point
TCP	Transmission Control Protocol
TUI	Terminal User Interface
UDP	User Datagram Protocol

VHDL	VHSIC Hardware Description Language
XNC	Extensible Network Controller
xDSL	x Digital Subscriber Line
xPON	x Passive Optical Networks
ZDT	Zitzler-Deb-Thiele

SEZNAM SYMBOLŮ

b	bit
B	bajt
G_i	graf
s	sekunda
ν	vrchol grafu
ε	hrana grafu
T_k	minimální kostra grafu
O	Omikron, asymptotická složitost
P	obor pravdivosti predikátové formule [výrokové formy]
ϕ	prázdná množina
t, T	reprezentant časové hodnoty
T_A	časová komplexnost
a, **b**, **c**, …	tučné malé písmeno – vektor
A, **B**, **C**, …	tučné velké písmeno – matice
∀	pro všechna, pro libovolné; obecný [velký] kvantifikátor, obecný [velký] kvantor, generalizátor
∈	<je> prvkem <množiny>, patří do <množiny> ; incidence [příslušnost] prvku k množině
$\mathbf{a}^T$	sloupcový [transponovaný] vektor
R_{-1}, R^{-1}	inverzní relace k relaci R
$\mathbf{A}_{(m,n)}$	matice **A** typu (m,n)
c_k	centroid s příslušností v k
DB	Daviesův-Bouldinův validační index
GHz	10^9 Hz, Hertz, jednotka frekvence (kmitočtu) v soustavě SI
SVD	Singulární rozklad matice

SEZNAM PŘÍLOH

A NetFlow verze 5 – vysvětlivky **104**

B Přehled evolučních algoritmů **105**

C Schéma genetického algoritmu FPGA **106**

D Obsah přiloženého CD **107**

A NETFLOW VERZE 5 – VYSVĚTLIVKY

Tab. A.1: NetFlow verze 5 – význam položek datagramu

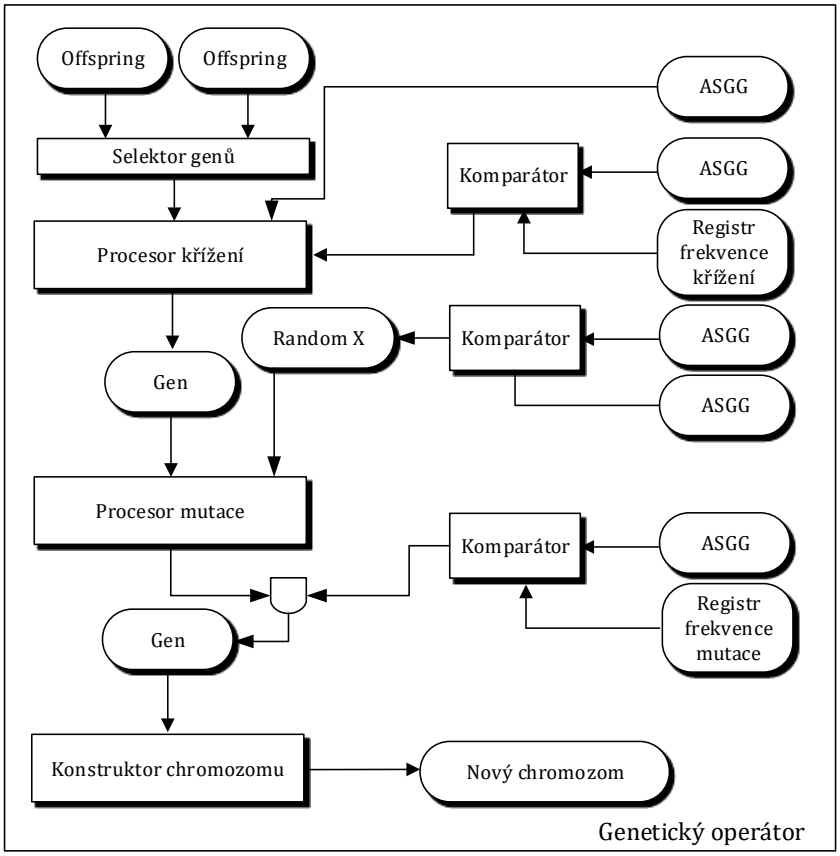
NetFlow hlavička		
Bajtů	**Název**	**Význam**
0–1	version	Verze NetFlow export formátu number
2–3	count	Počet exportovaných záznamů provozu (1-30)
4–7	sys_uptime	Aktuální čas v milisekundách od doby zapnutí zařízení
8–11	unix_secs	Aktuální čítač sekund od 0000 UTC 1970
12–15	unix_nsecs	Zbytkové nanosekundy od 0000 UTC1970
16–19	flow_sequence	Sekvenční čítač celkového počtu záznamů provozu
20	engine_type	Typ zařízení
21	engine_id	Číslo slotu
22–23	sampling_interval	První dva bity obsahují vzorkovací mód; ostatních 14 bitů hodnotu vzorkovacího intervalu
NetFlow zpráva		
0–3	srcaddr	Zdrojová IP adresa address
4–7	dstaddr	Cílová IP adresa address
8–11	nexthop	IP adresa následujícího uzlu router
12–13	input	SNMP index vstupu interface
14–15	output	SNMP index výstupu interface
16–19	dPkts	Počet paketů v exportovaném záznamu
20–23	dOctets	Celkový počet bajtů 3 vrstvy v paketech
24–27	first	Systémový čas v čase počátku zachycení záznamu
28–31	last	Systémový čas na konci zachycení záznamu
32–33	srcport	TCP/UDP zdrojové číslo portu
34–35	dstport	TCP/UDP cílové číslo portu
36	pad1	Nepoužito (nula) bajtů
37	tcp_flags	Kumulativní OR TCP návěstí
38	prot	Typ IP protocolu (například, TCP = 6; UDP = 17)
39	tos IP	Typ služby (ToS)
40–41	src_as	Číslo autonomního systému zdroje
42–43	dst_as	Číslo autonomního systému cíle
44	src_mask	Prefix masky zdrojové adresy
45	dst_mask	Prefix masky cílové adresy
46–47	pad2	Nepoužito (nula) bajtů

B PŘEHLED EVOLUČNÍCH ALGORITMŮ

Tab. B.1: Částečný výčet typů evolučních algoritmů

Název	Významní autoři	Princip
Genetické algoritmy	Holland, Michigan, 1975, USA	• binárně kódovaní jedinci • ruletová selekce • jednobodové křížení • mutace • inverze • teorie schémat
Evoluční strategie	Rechenberg, Schwefel, Berlín, 1965, Německo	• optimalizace vektorů reálných čísel • mutace je základní operací • většinou bez křížení
Evoluční programování	Owens, Walsh, Fogel, San Diego, 1962, USA	• spojitá optimalizace • evoluce konečných automatů • mutace • turnajové selekce • bez křížení
Genetické programování	Koza, Stanford, 1989 - 92, USA	• nelinerální stromová reprezentace (LISP) • evoluce programů - automatické generování programů • speciální operátory křížení, mutace a inicializace

C SCHÉMA GENETICKÉHO ALGORITMU FPGA



Struktura genetického operátoru FPGA

D OBSAH PŘILOŽENÉHO CD

Na přiloženém médiu jsou uloženy zdrojové kódy dizertační práce. Kód byl testován v Python 3.4.3 [MSC v.1600 64 bit (AMD64)] Microsoft Windows [Version 10.0.14393].

```
/ ..... kořenový adresář přiloženého CD
zdrojove_kody/
    ASGG/
    max_problem.py
    genetic.py
    benchmark_max_problem.py
    knapsack.py
    knapsack.csv
    genetic-algoritm
        K-MeansProfiler.pdf
        GeneticCore.py
        Genetics.py
        Vysledky-GA.txt
        K_Means.py
        data_1025.csv
        GeneticProfiler.pdf
        DaviesBouldin.py
        K_MeansReal.py
        data_1024.csv
        PDF
    gdp
        gdp-1.0.0.zip
    ripe_probe
        ripe_probe.py
    lifelines_test
        dataset.sqlite3
        database.py
        estimate_lifelines.py
```

PUBLIKAČNÍ ČINNOST

[A1] OUJEZSKÝ, V.; ŠKORPIL, V.; JURČÍK, M. Network Tomography Overview and Botnet Network Estimation, Part I. Access Server, 2015, roč. 13, č. 6, s. 1-4. ISSN: 1214-9675.

[A2] POLÍVKA, M.; OUJEZSKÝ, V.; ŠKORPIL, V. Modem Network Vulnerabilities and Security Testing – Actual Threats. In Proceedings of the 38th International Conference on Telecommunication and Signal Processing, TSP 2015. 1. Prague, Czech Republic: Asszisztencia Szervezo Kft., 2015. s. 33-36. ISBN: 978-1-4799-8497-8.

[A3] OUJEZSKÝ, V.; NOVOTNÝ, B.; Reliability and Availability Calculation for the Educational Laboratory. In Proceedings of the 21st Conference STUDENT EEICT 2015. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. s. 581-588. ISBN: 978-80-214-5148-3.

[A4] OUJEZSKÝ, V.; ŠKORPIL, V. Data Field Transformation from Ethernet Frame. International Journal of Emerging Research in Management and Technology, 2014, roč. 3, č. 4, s. 4-8. ISSN: 2278-9359.

[A5] OUJEZSKÝ, V.; ŠKORPIL, V. Cryptographic Sequence Generators for Stream Cipher and Their Behavioral Description. International Journal of Advanced Research in, Computer Science and Software Engineering, 2014, roč. 4, č. 3, s. 106-121. ISSN: 2277-128X.

Článek byl citován v:

- Volodymyr Maksymovych. *Poisson pulse sequence generators based upon modified Geffe generators*, Polytechnika Krakowska, DOI: 10.4467/2353737XCT.14.054.3962.

[A6] ŠKORPIL, V.; OUJEZSKÝ, V. Služby telekomunikačních sítí. Brno: VUT Brno, 2014. s. 1-130.

[A7] OUJEZSKÝ, V. Videokonferenční technologie v praxi - Vidět a slyšet na dálku. Upgrade IT!, 2008, roč. IV., č. 1/ 2008, s. 44-45. ISSN: 1801-5026.

[A8] OUJEZSKÝ, V.; HORVÁTH, T.; ŠKORPIL, V. Modeling Botnet C&C Traffic Lifespans from NetFlow Using Survival Analysis. In Proceedings of the 39th International Conference on Telecommunication and Signal Processing, TSP 2016. International Conference on Telecommunications and Signal Processing (TSP). Vienna, Austria: 2016. s. 50-55. ISBN: 978-1-5090-1287-9. ISSN: 1805-5435.

[A9] OUJEZSKÝ, V.; HORVÁTH, T. Case Study and Comparison of SimPy 3 and OMNeT++ Simulation. In Proceedings of the 39th International Conference on Telecommunication and Signal Processing, TSP 2016. International Conference on Telecommunications and Signal Processing (TSP). Vienna, Austria: 2016. s. 15-19. ISBN: 978-1-5090-1287-9. ISSN: 1805-5435.

[A10] OUJEZSKÝ, V.; HORVÁTH, T. NetFlow Console Collector–-Analyzer Developed in Python Language. In International Interdisciplinary PhD Workshop 2016. Brno: Brno University of Technology, Antonínská 548/1, Brno 601 90, 2016. s. 107-110. ISBN: 978-8-0214-5387-6.

[A11] OUJEZSKÝ, V.; HORVÁTH, T.; ŠKORPIL, V. Botnet C&C Traffic and Flow Lifespans Using Survival Analysis. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, 2017, roč. 6, č. 1, s. 38-44. ISSN: 1805-5443.

[A12] HORVÁTH, T.; OUJEZSKÝ, V.; MÜNSTER, P.; VOJTĚCH, J.; HAVLIŠ, O.; SIKORA, P. Modified GIANT Dynamic Bandwidth Allocation Algorithm of NG- PON. Journal of Communications Software and Systems, 2017, roč. 13, č. 1, s. 15-22. ISSN: 1845-6421.

[A13] OUJEZSKÝ, V.; HORVÁTH, T. Traffic Analysis Using NetFlow and Python. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 2017, ISSN: 2391-6761.

[A14] OUJEZSKÝ, V.; HORVÁTH, T.; ŠKORPIL, V. Traffic Similarity Observation by Using a Genetic Algorithm. 2017. Toho času odevzdáno k recenzi.

VEDENÉ ZÁVĚREČNÉ PRÁCE

Vedení závěrečných prací nebylo povinnou součástí, ale i z vlastního zájmu byla vypsána a vedena následující práce.

Monitorování provozu sítě pomocí dlouhodobě pracujícího analyzátoru

Tato diplomová práce se věnovala monitorování sítí a využití analyzátoru EndaceProbe s analytickou aplikací EndaceVision. V rámci práce byl popsán princip programovacího jazyka WSDL a protokolu SOAP. V praktické části byly vytvořeny tři laboratorní úlohy seznamující s praktickým využitím detekcí provozu. Součástí je také zapojení vysokorychlostního generátoru IXIA a směrovačů Cisco. Na těchto směrovačích bylo využito funkce SPAN pro zrcadlení provozu směrem k EndaceProbe. Ve vývojovém prostředí Netbeans IDE byl rozpracován koncept webových služeb.

Následující práce vedena nebyla. Z větší části byly poskytnuty konzultace.

Analýza vysokorychlostních sítí zátěžovým testerem

Tato bakalářská práce se zabývala návrhem laboratorních úloh pro praktické zvládnutí obsluhy vysokorychlostního generátoru provozu IXIA a TCL skriptů. Popisuje i teoretické znalosti z oblasti konfigurace přepínačů a směrovačů.

PROFESNÍ ŽIVOTOPIS

Zaměstnání

2016–doposud	IBM Client Inovation Center Central Europe
2017–doposud	Vysoké učení technické v Brně
2013–2016	T-Mobile Czech Republic a.s.
2006–2013	T-System Czech Republic a.s.

Studium

2009–2011	Vysoké učení technické v Brně – obor Telekomunikační a informační technika
2006–2009	Vysoké učení technické v Brně – obor Teleinformatika

Účast na projektech

2017	VI2VS/428 Detekce bezpečnostních hrozeb na aktivních prvcích kritických infrastruktur, MV ČR
2017	VI2VS/422 Redukce bezpečnostních hrozeb v optických sítích, MV ČR

Školní a pedagogické aktivity

2013–2017	Správa laboratoře transportních sítí centra SIX
2017	Člen IEEE
2014	Recenze IEEE Manuscript TCAD
2014	Oponentura diplomové práce
2015	Vedení diplomové práce

Certifikace, testy

CCNA	Cisco Certified Network Associated
CCNP	Cisco Certified Network Professional

Školení

Cisco ICDN1, ICDN2, A1, A2, R2, S1
HP & ComWare Configuration
Business Management (AKAD/IMAKA) (23440/92-34)
Linux LXI2-20120200015, LXI3 - 20130200011
SkillSoft Cisco Route, VHDL