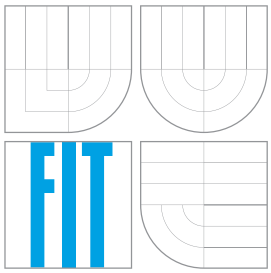
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ NÁVRH VYUŽÍVAJÍCÍ PŘEPISOVACÍ SYSTÉMY

EVOLUTIONARY DESIGN USING REWRITING SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BARBORA NÉTKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL BIDLO, Ph.D.

BRNO 2016

Zadání diplomové práce/17485/2015/xnetko00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Nétková Barbora, Bc.**
Obor: Bioinformatika a biocomputing
Téma: **Evoluční návrh využívající přepisovací systémy**
Evolutionary Design Using Rewriting Systems
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s problematikou evolučních algoritmů (EA).
2. Seznamte se s principy biologií inspirovaného vývinu, aplikovaného v evolučních algoritmech, kdy je tento proces realizován pomocí přepisovacích systémů (L-systémy, obecné gramatiky).
3. Zvolte vhodný typ přepisovacího systému a navrhněte jeho zakódování pro EA tak, aby bylo možné interpretovat jednotlivé symboly jako instrukce programovacího jazyka, funkční bloky či jiné vhodné entity související s řešeným problémem.
4. Navržený systém implementujte.
5. Zvolte alespoň dva problémy (např. z oblasti algoritmů, návrhu obvodových struktur apod.) a proveďte sadu experimentů, na kterých ukážete možnost jejich automatického (evolučního) návrhu s využitím techniky navržené v bodu 3.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Podle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bidlo Michal, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

Kotásek

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

V této práci byla navržena a implementována metoda pro evoluční návrh přepisovacích systémů. Pomocí genetického algoritmu jsou navrhována pravidla pro specifickou variantu Lindenmayerova systému. Navržené gramatiky jsou následně interpretovány jako rostoucí řadicí sítě. Byly prozkoumány různé přístupy interpretace L-systému na řadící sítě. Bude ukázáno, že evoluce je schopna navrhnout přepisovací systém pro částečně rostoucí sítě. Mezi nejlepší výsledky patří L-systémy navržené evolucí pro tvorbu sítí s 24 vstupy, které jsou schopny v dalších derivacích vytvořit síť až o 36 vstupech.

Abstract

This master's thesis proposes a method for the evolutionary design of rewriting systems. In particular, genetic algorithm will be applied to design rewriting rules for a specific variant of Lindenmayer system. The evolved rules of such grammar will be applied to generate growing sorting networks. Some distinct approaches to the rewriting process and construction of the sorting networks will be investigated. It will be shown that the evolution is able to successfully design rewriting rules for the proposed variants of rewriting processes. The results obtained exhibit abilities to successfully create partially growing sorting networks, which was evolved to grow for fewer inputs and in subsequent iterations grows up to 36 inputs.

Klíčová slova

Evoluční algoritmy, L-systémy, řadicí sítě, evoluční návrh, přepisovací systémy, rostoucí řadicí sítě, vývoj řadicích sítí po vstupech, vývoj řadicích sítí po vrstvách

Keywords

Evolutionary algorithms, L-systems, sorting networks, evolutionary design, rewriting systems growing sorting networks, development of sorting networks by inputs, development of sorting networks by layers

Citace

NÉTKOVÁ, Barbora. *Evoluční návrh využívající přepisovací systémy.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bidlo Michal.

Evoluční návrh využívající přepisovací systémy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana Ing. Michala Bidla Ph.D. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Barbora Nétková
24. května 2016

Poděkování

Děkuji svému vedoucímu Ing. Michalu Bidlovi Ph.D. za doporučení odborné literatury a cenné rady při tvorbě této práce. Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z Národního programu udržitelnosti II (NPU II) v rámci projektu IT4Innovations excellence in science - LQ1602.

Obsah

Kapitola 1

Úvod

V dnešní době se vývoj v informačních technologiích stále více prolíná s jinými obory. Dnes tak nejen informační technologie usnadňují práci v jiných oborech, ale i ostatní obory inspirují vývoj v informačních technologiích. Například biologie a živá příroda daly základ evolučním algoritmům a L-systémům, na kterých je postavena tato práce.

Evoluční algoritmy jsou založeny na Darwinově evoluční teorii. Jde o algoritmy optimalizovaně prohledávající prostor možných řešení. Při tomto prohledávání pak využívají operátory inspirované právě evolucí - mutace a křížení jedinců (možných řešení). Jedinci pro křížení, mutace, další generace apod. jsou vybírání podle ohodnocení. Vyšší ohodnocení znamená silnějšího a evolučně výhodnějšího jedince.

Lindenmayerovy systémy (L-systémy) jsou přepisovací systémy, původně navržené jako matematický model pro růst rostlin [15]. Oproti klasickým přepisovacím systémům se liší tím, že pravidla pro přepis znaků v řetězci jsou aplikována paralelně na všechny znaky v aktuálním řetězci. Vhodnou reprezentací mohou být L-systémy použity i pro návrh jiných struktur, například stolů [6], log [14] či jiných složitějších struktur [7].

Tato práce se věnuje právě navrhování struktur pomocí L-systémů, přičemž tyto L-systémy budou navrhovány pomocí evolučního algoritmu. Výsledný program implementovaný v rámci diplomové práce je schopen najít pomocí evolučního algoritmu L-systém, který generuje funkční řadicí síť. Cílem této práce je především nalézt L-systémy, které by po určitém počtu derivací byly schopny řadicí síť zvětšit, přičemž by tato síť byla stále plně funkční.

V úvodní kapitole této práce bude čtenář seznámen s principy a fungováním evolučních algoritmů. Následující kapitola pak pojednává o L-systémech a jejich variantách. Ve třetí kapitole jsou popsány řadicí sítě. Další kapitola pojednává o metodách konstrukce řadicích sítí pomocí L-systémů a rozdílnosti jednotlivých implementovaných verzí. V šesté kapitole je popsán implementovaný evoluční algoritmus. V následující kapitole jsou shrnuty experimentální výsledky a zobrazeny nejlepší nalezené L-systémy včetně sítí, které generují. Poslední kapitolou je závěr shrnující obsah práce, dosažené výsledky a možnosti dalšího výzkumu.

Kapitola 2

Evoluční algoritmy

Evoluční algoritmy (EA) jsou stále častěji používanou metodou stochastického optimalizovaného prohledávání prostoru řešení. Z počátku se používaly především k optimalizaci již existujících řešení, dnes už však jsou využívány i úplnému vyhledání řešení [17]. Tyto algoritmy jsou založeny na myšlence Darwinovy teorie, že přežít mohou jen nejlépe přizpůsobení jedinci. Pokud tedy budeme určitým způsobem reprodukovat kvalitní jedince, je pravděpodobné, že i jejich potomci budou kvalitní [11].

2.1 Základní pojmy EA

Než se seznámíme s průběhem a principy EA, je potřeba krátce vysvětlit několik základních pojmů. Některé z nich budou detailněji rozebrány v následujících podkapitolách.

- *Jedinec* je základní jednotkou EA, reprezentuje právě jedno vhodně zakódované řešení daného problému. Jeho součástí je obvykle soubor genů, reprezentující vlastnosti tohoto řešení a fitness hodnota reprezentující jeho kvalitu.
- *Fitness* je hodnota, která určuje jak kvalitní je jedinec. Pokud tato hodnota dosáhne maxima, jedinec s takovým ohodnocením je řešením daného problému. Zpravidla je tato hodnota vyčíslena uživatelem definovanou funkcí. Pokud je tato funkce navržena nevhodně, není zaručeno, že najdeme nejlepší řešení, či zda vůbec nějaké řešení nalezneme.
- *Populace* je množina (nebo multimnožina) aktuálních kandidátních řešení. Z jedinců v populaci v daném kroku (generaci) EA vybíráme ty nejlepší, s nimiž dále pracujeme. Počáteční populace bývá inicializována náhodně. Pokud však máme již nějaké znalosti o problému či známe nějaké (např. částečné či neoptimální) řešení, můžeme inicializaci počáteční populace optimalizovat pro náš problém.
- *Selekce* je genetický operátor sloužící pro výběr rodičovských jedinců z populace ať už pro křížení, mutaci či pro výběr jedinců do následující populace. Existuje mnoho možností, jak selekci provádět. Vždy se však snažíme vybírat jedince s vyšší hodnotou fitness, abychom zajistili co nejkvalitnější potomstvo/další generaci jedinců.
- *Reprodukce* tvorba nových jedinců z rodičovské populace pomocí tzv. genetických operátorů.

- Mezi *genetické operátory*, sloužící k tvorbě nových potomků, patří *křížení* a *mutace*. Pro křížení je vybráno několik jedinců z populace (rodiče). Z těchto rodičů jsou poté vytvořeni potomci specifickou kombinací genů z rodičovských jedinců. Každý potomek tedy musí mít minimálně 2 rodiče. Při mutaci je potomek tvořen z jednoho vybraného rodiče a to tak, že se náhodně změní hodnota některého genu.

2.2 Návrh EA

Při návrhu EA je nutné se nejprve důkladně zamyslet nad reprezentací řešení daného problému. Tuto reprezentaci nazýváme *genotyp*, to co genotyp reprezentuje a vyjadřuje nazýváme *fenotyp*. Vhodnou reprezentací zajistíme, že bude vyhledávání efektivní, a že budeme schopni nalezený výsledek správně interpretovat.

Dalším důležitým bodem návrhu je vhodně zvolená fitness funkce. Tato funkce nám ohodnotí jedince v populaci a na základě tohoto ohodnocení s jedinci dále pracujeme. Pokud je tedy tato funkce navržena a implementována nevhodně, jsou jedinci špatně ohodnoceni a výsledky budou zkreslené.

Dále musíme správně zvolit typy a pravděpodobnosti mutace a křížení a také způsob selekce.

Bezchybně fungující EA je vždy výsledkem déle trvajících experimentů, kdy musí uživatel jednotlivé části správně navrhnout, naprogramovat a poté doladit. Obvykle se dělají sady experimentů, v nichž se zkouší, jaká nastavení genetických operátorů jsou nejvhodnější, případně která fitness funkce lépe ohodnocuje daná řešení.

2.3 Průběh EA

Evoluční algoritmus může mít mnoho různých podob, to je dáno rozmanitostí druhů selekce a genetických operátorů. Pro potřeby této práce budeme uvažovat genetický algoritmus, který představuje jeden z nejrozšířenějších EA [5].

1. V první generaci je nejprve vytvořena počáteční populace, předem daného počtu jedinců. Jedinci v počáteční populaci mohou být vygenerováni buď náhodně, nebo to mohou být doposud známá či nedokonalá řešení problému.

2. Ohodnocení všech doposud neohodnocených jedinců populace.

3. Vyhodnocení ukončovacího kritéria. Pokud má nejlépe ohodnocený jedinec maximální hodnotu fitness je nalezeno řešení a algoritmus končí. Pokud není nalezen jedinec s maximální hodnotou fitness, algoritmus pokračuje. Algoritmus může být také ukončen po dosažení maximálního počtu generací, nebo jiným ukončovacím kritériem.

4. Výběr jedinců do rodičovské populace pro tvorbu potomků křížením a mutací.

5. Reprodukce - tvorba populace potomků z rodičovské populace. Aplikace genetických operátorů s určitou pravděpodobností.

6. Tvorba nové populace z předchozí populace, z populace potomků, případně i z nově náhodně vygenerovaných jedinců.

7. Inkrementace počítadla generací. Zpět k bodu 2.

2.4 Genetické operátory

Genetické operátory slouží k tvorbě potomků z rodičovské populace k tzv. reprodukci. Poté co je určitou metodou selekce vybrána rodičovská populace, jsou z této populace tvořeni potomci podobně jako je tomu v přírodě. Rozlišujeme dva základní operátory pro tvorbu nových jedinců a to křížení chromosomů a mutaci genu.

Aplikace operátorů mutace a křížení vždy probíhají s určitou pravděpodobností. Příliš vysoká pravděpodobnost nám velmi mění a narušuje již nalezená dobrá řešení. Příliš nízká pravděpodobnost naopak vede k nerozmanité populaci. Při ladění EA je tedy důležité pomocí experimentů tyto pravděpodobnosti nastavit co možná nejvhodněji.

V následujících sekcích jsou představeny základní typy genetických operátorů a jejich nejpoužívanější varianty. V reálných implementacích složitějších problémů je však většinou potřeba genetické operátory a jejich pravděpodobnosti vhodně přizpůsobit řešenému problému.

2.4.1 Selekce

Jak již bylo zmíněno, selekcí se rozumí výběr jedinců z populace. Všechny metody selekce se snaží vybírat do rodičovské populace dostatečně kvalitní jedince. Ne vždy však bývají vybíráni pouze ti nejlepší. Takový výběr by vedl k rychlé degradaci populace na sobě velmi podobné jedince, ztratila by se rozmanitost řešení a hrozilo by tak uváznutí v lokálním maximu fitness funkce. Pokud však naopak nebudeme vybírat dostatečně kvalitní jedince, algoritmus bude k hledanému řešení konvergovat jen velmi pomalu.

Mezi nejznámější a nejpoužívanější metody selekce patří *elitismus*, *ruleta* a *turnajový výběr*. Tyto metody můžeme v námi navrženém EA také kombinovat. Například pro mutaci zvolíme nejlepšího jedince (elitismus) a pro křížení provedeme výběr rodičů turnajem.

Následující sekce detailněji přibližují nejpoužívanější druhy selekce. Pro názorné příklady bude použita populace o 4 jedincích zobrazena v tabulce 2.1.

Jedinec	Fitness hodnota
A	20
B	10
C	5
D	2

Tabulka 2.1: Ukázková populace čtyř jedinců

Elitismus

Tato metoda výběru spočívá v seřazení všech jedinců podle jejich hodnoty fitness. Poté vybereme námi potřebné množství nejlepších jedinců. Při výběru 50% jedinců populace z tabulky 2.1 by tedy byli vybráni 2 nejlepší jedinci A a B.

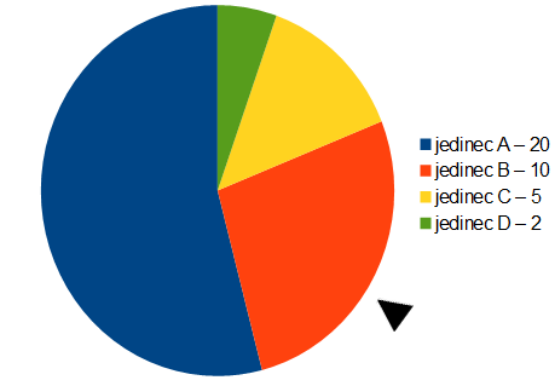
Ruleta

Při ruletovém mechanismu (také známý pod anglickým názvem *rulette wheel* nebo *fitness-propotrionate selection*) je pravděpodobnost výběru jedince přímo úměrná velikosti jeho fitness. Čím větší fitness jedinec má, tím větší je šance vybrání. Oproti předchozí metodě

mají jedinci s malou fitness hodnotou šanci na vybrání také, i když s velmi malou pravděpodobností. Tato metoda se řadí k nejpoužívanějším.

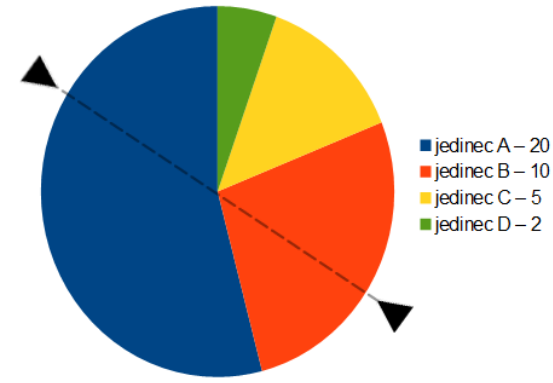
Nejprve je tedy nutné mít všechny jedince ohodnocené. Potom jim je v ruletovém kole přiřazena výseč odpovídající velikosti jejich fitness hodnoty. Toto ruletové kolo je následně roztočeno a po zastavení je vybrán jedinec, na jehož výseč ukazuje ukazatel (viz obrázek 2.1). Pravděpodobnost výběru daného jedince i v populaci o velikosti N je pak dána vztahem:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \quad i \in \{1, \ldots, N\} \tag{2.1}$$



Obrázek 2.1: Ruletové kolo pro výběr 1 jedince ze 4. Ukazatel vybral po ukončení otáčení jedince B s druhou nejvyšší fitness.

Pokud potřebujeme vybrat více jedinců, roztočíme ruletové kolo tolikrát, kolik jedinců je třeba vybrat. Nebo po okraji ruletového kola rozmístíme pravidelně tolik ukazatelů, kolik chceme jedinců (viz obrázek 2.2).



Obrázek 2.2: Ruletové kolo pro výběr 2 jedinců. Ukazatel vybral po ukončení otáčení jedince A a B.

Turnajový výběr

Turnajový výběr (anglicky známý jako *tournament selection*) je metoda selekce, při níž je náhodně vybráno několik jedinců, kteří se zúčastní tzv. turnaje. V turnaji poté vítězí jedinci

s vyšší hodnotou fitness. Tato metoda opět dává šanci i jiným než nejlepším jedincům dostat se do rodičovské populace a to právě náhodným výběrem do turnaje. Soutěžení jedinců v samotném turnaji nám pak obvykle zajistí dostatečně kvalitní jedince.

Nejčastější formou turnaje je náhodné vybrání dvou jedinců z populace, jejich vyhodnocení a porovnání. Z takového turnaje tedy vzejde jeden vítěz. Pokud bychom v tomto případě potřebovali vítězů více, turnaj budeme opakovat.

Když například potřebujeme vybrat 2 rodiče z tabulky 2.1, uděláme dvakrát turnajový výběr. Například, v prvním turnaji budou soutěžit jedinci A a C, ve druhém turnaji jedinci C a D. Z těchto dvou turnajů pak tedy vzejdou dva vítězové a to jedinec A ($fitness(A) > fitness(C)$) a jedinec C ($fitness(C) > fitness(D)$) .

2.4.2 Mutace

Mutace pobíhá zpravidla na jednom náhodně vybraném genu rodičovského jedince. Tento náhodně vybraný gen je pak náhodně změněn. V biologii můžeme mutaci pozorovat například při kopírování DNA. Tento jev probíhá obvykle s velmi malou pravděpodobností např. $p_m = 0.1\%$ [17].

Nejjednodušším příkladem je mutace na binárně zakódovaném genu, kde dojde k inverzi binární hodnoty. Viz následující příklad v tabulce 2.2.

Rodič:	1100111000
Potomek:	1100111100

Tabulka 2.2: Ukázka mutace

Pokud může gen nabývat více hodnot, vybereme náhodně jednu z přípustných. Při reprezentaci užívající reálná čísla pak například gen můžeme vynásobit náhodnou hodnotou či jinak hodnotu upravit [8]. Složitější jsou pak mutace v souborech genů, které mají nějaké omezující podmínky (například jde o permutace v úloze obchodního cestujícího). V takovýchto chromosomech je nutné po změně jednoho genu ještě upravit ostatní, aby byl celý chromosom i po provedení mutace validní.

2.4.3 Křížení

Při operaci křížení probíhá tvorba nového jedince specifickým sloučením rodičovských chromosomů. Z pravidla má potomek rodiče minimálně dva a každý z nich mu předá nějakou genetickou informaci. I tento operátor byl inspirován biologickým procesem, můžeme ho pozorovat například při meiotickém dělení buněk. Tento jev probíhá s vysokou pravděpodobností okolo 70% [17]. Genetická informace totiž není pouze náhodně změněna, ale kombinuje již dvě existující řešení.

Existuje mnoho způsobů křížení. Který typ křížení užijeme, závisí na kódování chromosomu a jeho struktuře. Mezi nejznámější typy křížení patří *jednobodové* a *vícebodové křížení*, *uniformní křížení*. Zvláštní skupinou jsou pak typy křížení pro reálná čísla a chromosomy s omezujícími podmínkami jako jsou permutace, již zmíněné u mutací.

K-bodové křížení

Tento typ křížení spočívá v rozdělení chromosomu každého rodiče K náhodně určenými body na $K + 1$ úseků. Chromosomy potomků jsou pak složeny střídavě z úseků jednoho či

druhého rodiče. Nejjednodušším příkladem takového křížení je křížení jednobodové ($K = 1$). Viz tabulka 2.3, ukázku vícebodového křížení pak můžete vidět v tabulce 2.4

Rodiče:	1101001101	1011010011
Potomci:	1101010011	1011001101

Tabulka 2.3: Ukázka jednobodového křížení

Rodiče:	1101001101	1011010011
Potomci:	1101010101	1011001011

Tabulka 2.4: Ukázka dvoubodového křížení

Uniformní křížení

Při uniformním křížení je rozhodnuto pro každý gen zvlášť, z kterého bude rodiče a to s pravděpodobností $p_u = 0.5$". Názorná ukázka je zobrazena v tabulce 2.5.

Rodiče:	1101001101	1011010011
Potomci:	1011011001	1101000111

Tabulka 2.5: Ukázka uniformního křížení

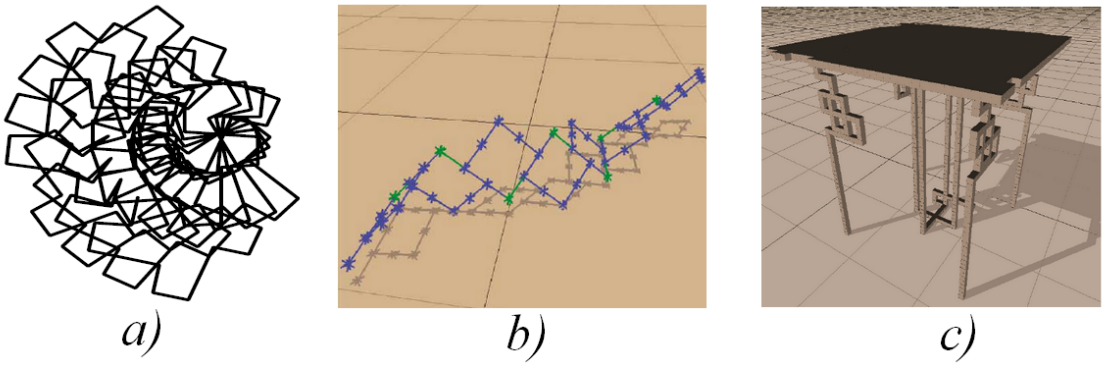
Ostatní typy křížení

Specifické typy kódování si žádají specifické křížení, proto tak například u reprezentací s reálnými čísly probíhá křížení například aritmetickým průměrem hodnot genů rodičů nebo odmocninou součinu genů rodičů [8]. U chromosomů s omezujícími podmínkami je opět třeba myslet na dodatečnou úpravu chromosomů potomků tak, aby zůstali validní.

Kapitola 3

L-systémy

Cílem této práce je využít výše popsaný evoluční algoritmus k evolučnímu návrhu přepisovacího systému. Zvláštní skupinou přepisovacích systémů hojně využívaných ve výpočetním developmentu jsou Lindenmayerovy systémy (zkráceně L-systémy) [12], [13]. Byly navrženy v roce 1968 biologem Aristidem Lindenmayerem jako matematická teorie vývoje jednoduchých rostlin. Postupem času si však našli využití i v jiných oblastech, zejména při evolučním návrhu např. stolů [6], log [14] či jiných mechanických struktur [7]. Ukázky těchto struktur můžete vidět na obrázku 3.1.



Obrázek 3.1: Ukázka struktur generovaných L-systémy a) loga [14], b) pohybující se stvoření [7], stoly [6]

Tyto systémy se od ostatních přepisovacích systémů, jak je známe z Chomského hierarchie, liší především paralelním přístupem. To je dáno tím, že byly inspirovány biologickým vývojem. Jejich základním prvkem jsou symboly reprezentující jednotlivé buňky organismu [17]. Všechny tyto buňky se pak v průběhu času vyvíjejí současně, tedy i v L-systému se všechny symboly přepisují paralelně.

Existuje mnoho variant a typů L-systémů, nejjednodušším je D0L-systém popsaný v následující podkapitole. Dále existuje několik rozšiřujících typů L-systémů, jako jsou například L-systémy s podmínkou, zakazující, parametrické či stochastické.

Evolučně se navrhují zejména samotná pravidla L-systému, u složitějších typů můžeme pomocí evoluce hledat také podmínky a parametry.

3.1 D0L-systémy

Tím nejjednodušším typem L-systému je tzv. D0L-systém. Je to systém deterministický a bezkontextový. Formální definicí [15] DOL systému je triplet $G = (V, \omega, P)$, kde:

- V je konečná abeceda, množina obsahující všechny symboly L-systému,
- V^* je množina všech řetězců, které můžeme vytvořit nad abecedou V a V^+ je množina všech neprázdných slov,
- $\omega \in V^+$ je počáteční řetězec, tzv. axiom.
- $P \subset V \times V^+$ je konečná množina pravidel. Pravidlo $(a, \chi) \in P$ zapisujeme ve tvaru $a \to \chi$, kde $a \in V$ je předchůdce (*predecessor*) a $\chi \in V^+$ následník (*successor*).

Na začátku výpočtu tedy známe axiom, abecedu symbolů a pravidla. Axiom je v průběhu výpočtu neustále přepisován aplikací pravidel. Na všechny symboly aktuálního řetězce jsou v každém kroku paralelně aplikována odpovídající pravidla.

Jedním z nejznámějších příkladů tohoto systému je vykreslení fraktálu nazývaného *Kochova křivka*, jejíž vývin můžete vidět na obrázku 3.2. Její variací je *Kochova vločka* na obrázku 3.3, která se liší pouze rozšířeným axiomem. Abeceda tohoto L-systému obsahuje tři symboly F, $+$, a $-$, které mají následující význam:

- F: nakresli úsečku,
- $+$: změň směr vykreslování úsečky o 60° proti směru hodinových ručiček,
- $-$: změň směr vykreslování úsečky o 60° po směru hodinových ručiček.

Axiomem *Kochovy křivky* je symbol F a množina přepisovacích pravidel obsahuje tato pravidla:

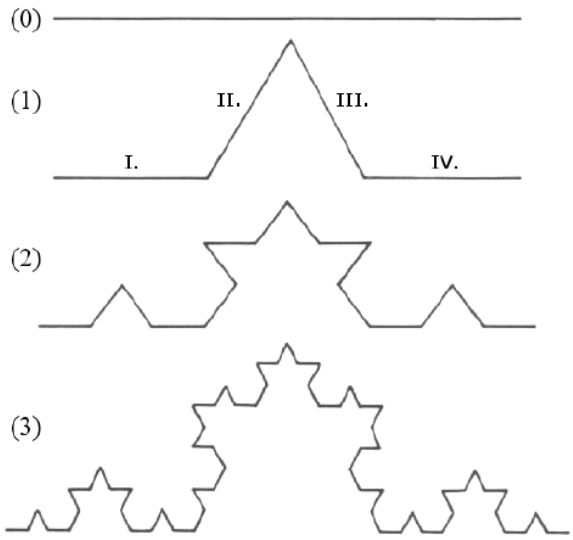
1. $F \to F + F - -F + F$
2. $+ \to +$
3. $- \to -$

3.2 Složitější typy L-systémů

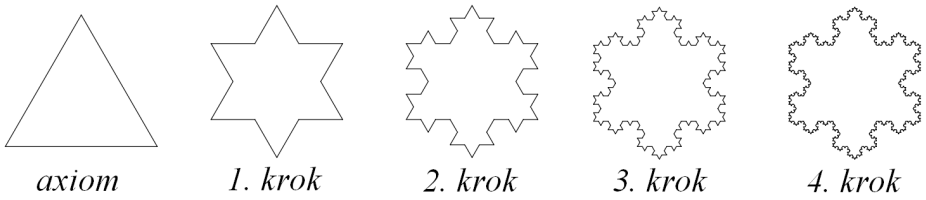
Složitější typy L-systémů se od základního liší zpravidla změnou podoby pravidel či v určitém omezení jejich aplikace. Pro výpočetní development jsou nejpoužívanější L-systémy podmínkové a parametrické.

3.2.1 Podmínkové a zakazující L-systémy

Podmínkové/zakazující L-systémy jsou takové L-systémy, kde ke každému pravidlu je přiřazena množina řetězců, které povolují/zakazují užití daného pravidla, pokud je tento řetězec podřetězcem v aktuálním derivačním kroku.



Obrázek 3.2: První 4 kroky vývoje fraktálu Kochovy křivky pomocí L-systému.



Obrázek 3.3: První 4 kroky vývoje fraktálu Kochovy vločky pomocí L-systému.

3.2.2 Parametrické 0L-systémy

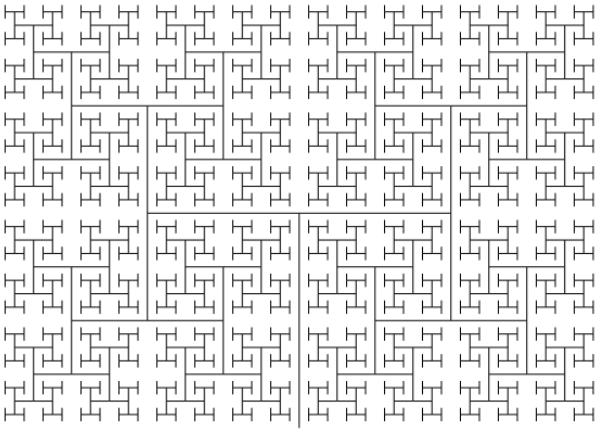
Pravidla v parametrických 0L-systémech (P0L-systémech) se liší tím, že se symboly v pravidlech jsou spojeny parametry, které udávají např. délku vykreslené úsečky, nebo se na ně pojí podmínka. Takovýmto parametrem může být např. reálné číslo, proměnná, aritmetický výraz nebo i funkce, jejíž hodnota je vyčíslena až za běhu. Pro zápis parametrů se užívá závorkové syntaxe. Příkladem takového pravidla je např.: $A(t) \rightarrow B(t+1)$, kde t je parametr.

Pravidla parametrických systémů v sobě obsahují také podmínky a jejich obecný tvar lze zapisovat ve tvaru oddělujícím předchůdce, podmínku a následníka symboly : a $\rightarrow$. Obecný tvar je tedy *předchůdce*: *podmínka* $\rightarrow$ *následník*. Příkladem takového pravidla je např. $A(t) : t > 0 \rightarrow B(t+1)CD(t^{0.5}, t-2)$.

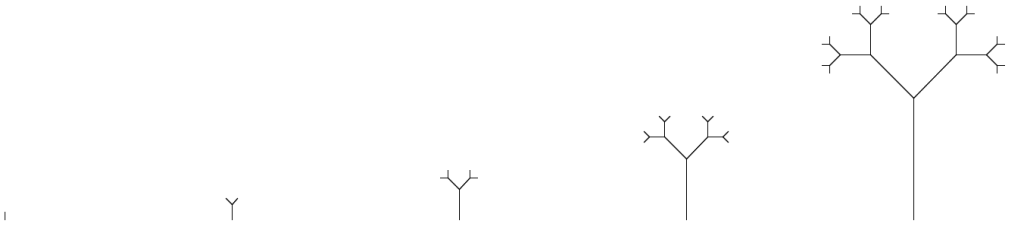
Formálně jsou P0L-systémy definovány [15] čtveřicí $G = (V, \sum, \omega\ P)$, kde

- V je abeceda systému,
- $\sum$ je množina parametrů,
- $\omega \in (V \times \mathbb{R}^*)^+$ je neprázdné parametrické slovo (axiom),
- $P \subset (V \times \sum^*) \times C(\sum) \times (V \times \varepsilon(\sum))^*$ je konečná množina pravidel.

Jednoduchým příkladem parametrického systému je fraktál *H-strom* na obrázku 3.4. Abecedou tohoto stromu jsou symboly $F, H, +, -, [,]$ axiomem je $F(200)$. Přepisovací pravidlo je následující: $F(t) \rightarrow H(l)[+F(l * R)][-F(l * R)]$. R a U jsou proměnné L-systému a pro jejich různě nastavené hodnoty se výsledek mírně liší. R je proměnná určující délku vykreslené úsečky (proměnná pro symbol F). U je proměnná určující úhel otočení (proměnná pro symboly $+$, $-$). Pro ukázkový H-strom na obrázku 3.4 jsou nastaveny na $R = \frac{1}{\sqrt{2}}$ a $U = 90°$. Název H-strom je odvozen od vzhledu, který připomíná písmena H. Jiná varianta H-stromu s parametry $R = \frac{1}{\sqrt{2}}$ a $U = 45°$ je tzv. *Pythagorův strom* na obrázku 3.5 můžete vidět jeho první 4 derivace.



Obrázek 3.4: Fraktál H-strom s parametry $R = \frac{1}{\sqrt{2}}$ a $U = 90°$



Obrázek 3.5: První 4 kroky vývoje fraktálu Pythagorova stromu pomocí L-systému s parametry $R = \frac{1}{\sqrt{2}}$ a $U = 45°$

1L-systémy obsahují pravidla s kontextem pouze na jedné straně - pravé nebo levé, *2L-systémy* obsahují pravidla, která mají kontext na obou stranách. Pravidla v 1L i 2L-systémech mohou obsahovat i pravidla bez kontextu, pravidla obsahující kontext však mají přednost při aplikaci. Existují také parametrické kontextové systémy.

3.3 Interpretace L-systémů

Grafické vizualizaci, kterou jste mohli vidět na všech dosud uvedených příkladech L-systémů, se říká *Želví grafika*. Tato interpretace L-systémů je jedna z nejjednodušších a proto také bývá často využívána i při developmentu ať už objektů ve 2D [14] nebo 3D prostoru [6].

Symboly v řetězci vygenerované L-systémem jsou interpretovány jako příkazy pro želvu

nesoucí štětec. Stav želvy je dán pozicí v souřadnicovém systému a orientací želvy, případně můžeme ještě definovat druh štětce.

Základními operacemi, které želva vykonává, jsou:

- posun z bodu A do bodu B s vykreslením úsečky AB,
- posun z bodu A do bodu B bez vykreslení úsečky,
- otočení o úhel α po směru hodinových ručiček a
- otočení o úhel α po směru hodinových ručiček.

Pokročilými operacemi jsou například uložení aktuálního stavu želvy do zásobníku a pozdější využití tohoto uloženého stavu nebo změna štětce želvy.

Želva však kromě kreslení čar může například vkládat pixely či jejich analogii ve 3D (voxely), jako je tomu například v experimentu navrhujícím pomocí L-systémů stoly [6]. V tomto experimentu byly výsledné řetězce interpretovány tak, že želva pohybující se prostorem — 3D maticí prázdných voxelů — vždy vyplnila voxel, do kterého vstoupila.

3.4 Úskalí a výhody evolučního návrhu L-systémů

Jak již bylo zmíněno, evolučně se navrhují většinou přímo jednotlivá pravidla pro L-systém, nebo můžeme pomocí evoluce například jen hledat vhodné parametry již známých pravidel.

Při evolučním návrhu je všeobecně problém vhodně navrhnout systém a zakódovat náš problém. U evolučního návrhu pravidel se však můžeme setkat s další překážkou a tou je ohodnocení L-systému. Systém jako takový hodnotíme totiž fitness funkcí jako celek většinou po vhodné interpretaci vygenerovaného řetězce. Takovou fitness funkcí může být vizuální stránka [14], u návrhu stolů stabilita a množství materiálu [6] nebo u řadicích sítí počet správně seřazených posloupností. Tím, že však hodnotíme L-systém jako celek, nemáme informaci o kvalitě jednotlivých pravidel ale o celém setu pravidel užitých v tomto systému. Tak se může snadno stát, že výsledky kvalitních pravidel budou při vývinu přepsány jinými pravidly, která nedávají tak dobré výsledky [10]. Výsledný systém tedy ohodnotíme nižší fitness hodnotou a může se stát, že jedince s několika kvalitními pravidly při evoluci ztratíme.

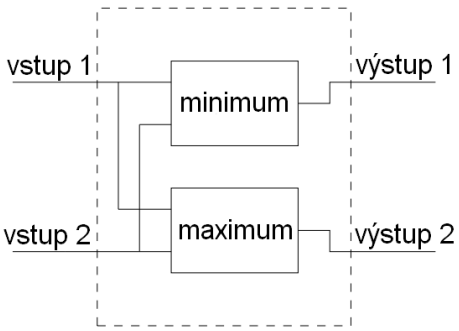
Dalším problémem evolučního návrhu je škálovatelnost, tedy to, že s rostoucí složitostí problému (např. rostoucí počet vstupů v řadicí síti) roste prostor prohledávaných řešení. Právě toto by mohlo být překonáno pomocí L-systémů, které jsou schopny z axiomu pomocí pravidel stále zvětšovat generovaný řetězec a tím rozšiřovat řešení. Pokud je součástí L-systému axiom, který je počátečním jednoduchým řešením daného problému, aplikací pravidel bychom pak mohli tento axiom rozšířit a vytvořit tak větší řadicí síť. Tomuto přístupu tvorby řešení z počátečního embrya se říká development a byl využit při návrhu řadicích sítí např. v [2] a [3].

Kapitola 4

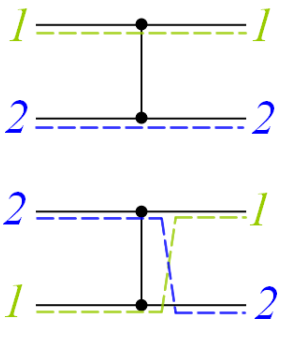
Řadicí sítě

Pokud chceme evolučně navrhovat L-systémy, musíme mít ujasněno, jaký má být výsledek interpretace výsledného řetězce vygenerovaného L-systémem, tedy co vlastně chceme evolučně navrhnout. Tato práce se věnuje evolučnímu návrhu řadicích sítí za použití L-systémů. Tato kapitola se proto věnuje vysvětlení základních principů a konstrukcí řadicích sítí.

Řadicí sítě byly představeny v roce 1954. Historii a problematiku řadicích sítí popsal Donald Knuth v své publikaci [9]. Řadicí síť se skládá ze dvou typů prvku a to z vodičů a komparátorů. Komparátor je součástka se dvěma vstupy x a y a dvěma výstupy x' a y', která provádí operaci *compare and swap*. Pokud komparátor řadí vzestupně na výstupu x' bude minimum a na výstupu y' bude maximum ze dvou vstupních hodnot. Schéma komponenty vzestupného komparátoru je zobrazeno na obrázku 4.1 a na obrázku 4.2 můžeme vidět jednoduchou ukázku seřazení dvou prvků jedním takovýmto komparátorem.

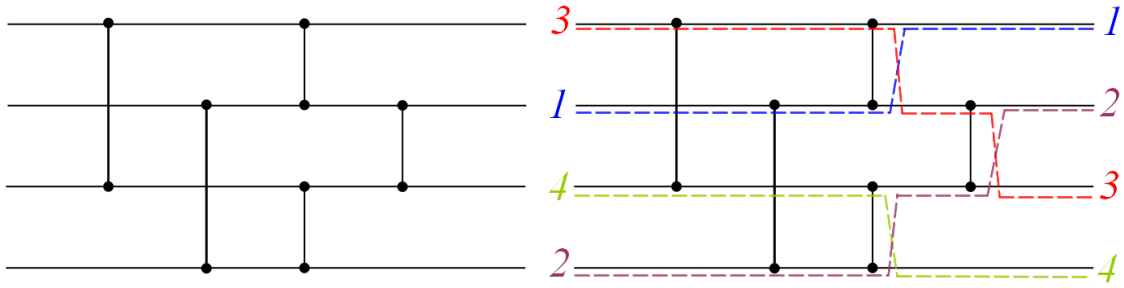


Obrázek 4.1: Schéma komponenty komparátoru.



Obrázek 4.2: Ukázka vzestupného seřazení dvouvstupé sítě s jedním komparátorem.

Tyto komparátory poté jsou ve větší síti vhodně rozloženy tak, aby výstupem byla seřazená posloupnost všech možných kombinací vstupních prvků. Rozložení těchto komparátorů není závislé na vstupní sekvenci, ale pouze na počtu vstupů. Právě pro tuto neměnnost jsou řadicí sítě vhodné k paralelizaci a hardwarové implementaci. Na obrázku 4.3 je zobrazeno schéma řadicí sítě pro 4 vstupy a názorná ukázka seřazení sekvence pomocí pěti vhodně rozložených komparátorů.



Obrázek 4.3: Ukázka čtyřvstupé sítě s pěti komparátory a jejího řazení.

4.1 Komponenty řadicích sítí

Pro potřeby této práce nazveme každý vstup řadicí sítě **vodičem**. Například čtyřvstupá řadicí síť se tedy bude skládat ze čtyř samostatných vodičů. Pokud bychom chtěli rozšířit takovou síť na šestivstupou přidali bychom 2 vodiče.

Samotný vodič se skládá z několika **komponent**, z nichž každá má specifický význam. V této práci budou použity 4 následující typy komponent:

1. **Vstup (I)** je komponenta na začátku každého vodiče.
2. **Výstup (O)** je komponenta na konci každého vodiče.
3. **Komparátor (C)** je komponenta na vodiči (ani na začátku, ani na konci), která je schopná operace *compare and swap*. Tato komponenta spojuje aktuální vodič s jiným vodičem podle šířky komparátoru.
4. **Drát (W)** je komponenta na vodiči (ani na začátku, ani na konci), která neobsahuje komparátor. Jde v podstatě o prázdný drát, na který se případně může připojit komparátor z jiného vodiče.

Tyto komponenty jsou důležité pro interpretaci řetězce vygenerovaného L-systémem na řadicí síť. Každý vodič začíná vstupem, následuje libovolná posloupnost drátů a komparátorů, a poté je vodič ukončen výstupem. Viz regulární výraz 4.1.

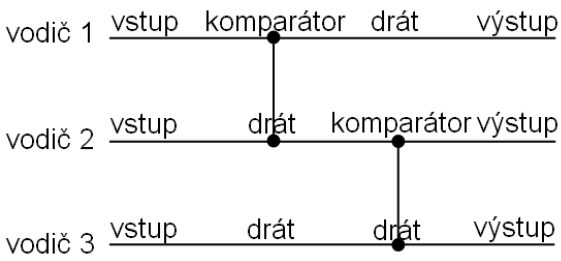
$$I(C,W)^*O \tag{4.1}$$

Na obrázku 4.4 můžete vidět jednoduchou řadicí síť s označením jednotlivých vodičů a komponent.

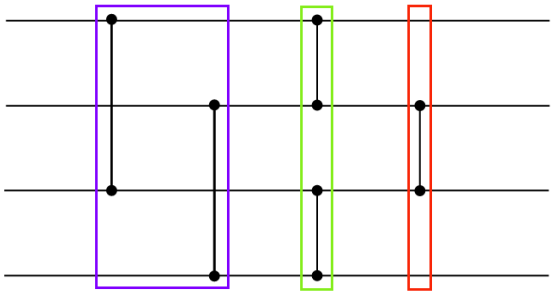
4.2 Parametry řadicích sítí

Řadicí sítě mají 2 hlavní parametry, podle kterých je můžeme optimalizovat. Prvním je *počet použitých komparátorů*. Druhým je zpoždění, které se rovná počtu vrstev. Vrstva řadicí sítě je skupina nezávislých komparátorů, takovéto komparátory nesmí mít žádný společný vstup ani výstup aby si vzájemně neovlivňovaly výsledky. Na obrázku 4.5 můžeme vidět jednotlivé vrstvy sítě z předchozího obrázku 4.3 označené různými barvami.

Při návrhu sítí se snažíme jeden (nejlépe oba) z těchto parametrů minimalizovat. Jednou z metod minimalizace je odstranění redundantních komparátorů. *Redundantní komparátor*



Obrázek 4.4: Třívstupá síť s označením jednotlivých vodičů a komponent



Obrázek 4.5: Čtyřvstupá síť s pěti komparátory a třemi vrstvami.

nikdy nezpůsobí výměnu hodnot na svých vstupech a proto je v síti zbytečný. Takovým komparátorem může být například komparátor připojený na stejné vstupy jako komparátor předchozí.

Pro nízké hodnoty vstupů ($n \leq 8$) již byly různými metodami nalezeny optimální (minimální) řadicí sítě. Přehled parametrů nejlepších známých řadicích sítí je zobrazen v tabulce 4.2.

Počet vstupů	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Zpoždění	0	1	3	3	5	5	6	6	7	8	8	9	10	10	10	10
Počet komparátorů	0	1	3	5	9	12	16	19	25	29	35	39	45	51	56	60

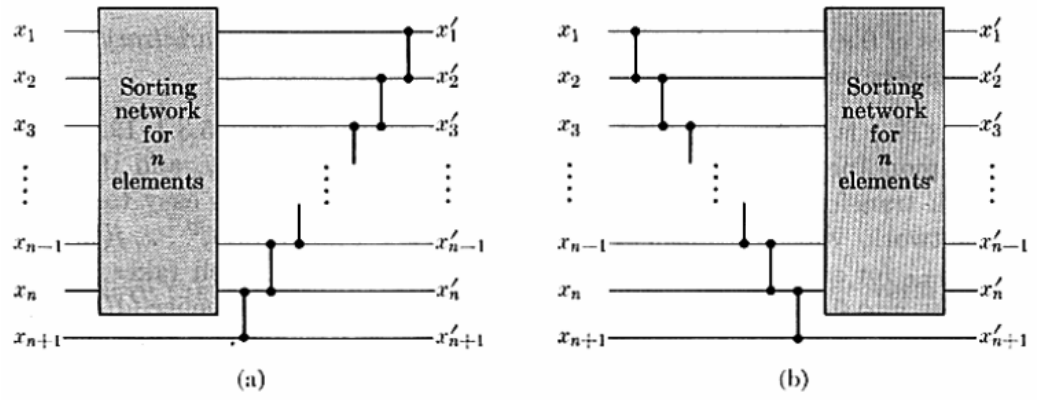
Tabulka 4.1: Doposud nejlepší známé parametry řadicích sítí. [9]

4.3 Konvenční metody návrhu

Jak již bylo řečeno řadicí sítě obvykle navrhujeme pro pevný počet vstupů se snahou minimalizovat zpoždění a počet komparátorů. Takový návrh však pro velké počty vstupů není vždy snadný, rychlý a někdy ani možný. Proto, pokud nepotřebujeme v praxi síť, která má parametry optimalizované, můžeme k rychlému návrhu využít jednu některou z generických metod konstrukce řadicích sítí. Například princip vkládání (*insert-sort*), princip výběru (*select-sort*), případně pokročilejší metody [16], [1].

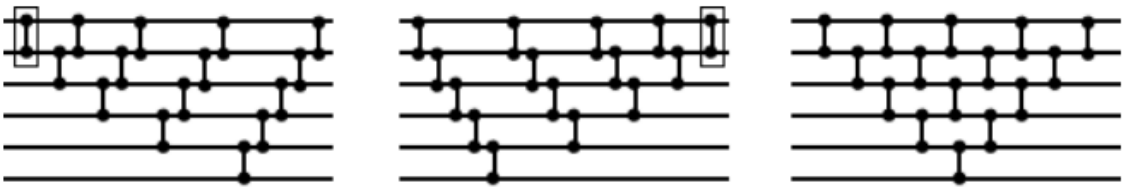
Insert-sort i select-sort rozšiřují již fungující n-vstupou síť vždy o 1 vstup (obrázek 4.6). Při vkládání se nejprve seřadí n vstupů původní sítě a až poté je zařazena hodnota ze vstupu $n + 1$ na správné místo. Při výběru je nejprve na vstup $n + 1$ zařazena nejvyšší hodnota a zbylých n vstupů je poté seřazeno původní sítí. Tyto techniky se obvykle využívají

k rozšíření nějak optimalizované sítě, ne k návrhu sítě od úplného počátku. Protože počet komparátorů a zpoždění nejsou ani zdaleka optimální, výhodou tohoto řešení je však jeho škálovatelnost.



Obrázek 4.6: Konstrukce sítě s $n + 1$ vstupy: a) vkládáním b) výběrem [9]

Pokud bychom přece jen využili tyto techniky postupného zvětšování sítě již od začátku návrhu zjistili bychom, že po vhodném posunutí komparátorů, tak aby mohly pracovat paralelně, získáme dvě totožné sítě viz obrázek 4.7.



Obrázek 4.7: Konstrukce sítě s $n+1$ vstupy: a) vkládáním b) výběrem c) paralelní zpracování předchozích [9]

4.4 Ověření správnosti

Při návrhu nové sítě je samozřejmě důležité, aby výsledná síť fungovala správně, tedy správně seřadila libovolnou posloupnost čísel přivedenou na vstupy. Pro příklady malých sítí z předchozích obrázků, je formální důkaz triviální.

S rostoucím počtem vstupů, se takový důkaz stává nemožným a je potřeba dokázat, že bude správně seřazeno všech $n!$ kombinací vstupních hodnot. Počet testovaných posloupností však roste faktoriálně v závislosti na vstupech a tak ani touto metodou obvykle nejsme schopni námi vytvořenou síť verifikovat.

Optimalizací předchozího přístupu je *zero-one princip*, který říká, že pokud je síť schopna správně seřadit všech 2^n vstupních kombinací 0 a 1, pak je schopna seřadit jakoukoli posloupnost. Důkaz tohoto principu je uveden v [9]. Pomocí zero-one principu tedy rapidně snížíme počet vstupních posloupností, které je potřeba ověřit. Přesto je však složitost takové evaluace sítí exponenciální.

Kapitola 5

Konstrukce řadicích sítí pomocí L-systemů

L-systémy byly původně vymyšleny pro modelaci růstu rostlin. To bylo také inspirací pro návrh algoritmu, jehož jednotlivé varianty byly posléze implementovány a experimentálně otestovány. Jednotlivé vodiče si můžeme představit jako části rostliny, které v průběhu vývoje rostou (přidání vodiče bez komparátoru) nebo se větví (přidání komparátoru).

V této práci budou představeny dva přístupy pro tvorbu řadicích sítí pomocí přepisovacího systému. Tyto přístupy se liší pouze interpretací výsledného řetězce generovaného L-systémem. Prvním přístupem je tvorba sítě postupně po jednotlivých vodičích, druhým případem je „více konvenční“ tvorba sítě po jednotlivých vrstvách. Obě varianty interpretace jsou detailněji popsány v sekcích 5.3 a 5.4.

5.1 L-systém navržený pro tuto práci

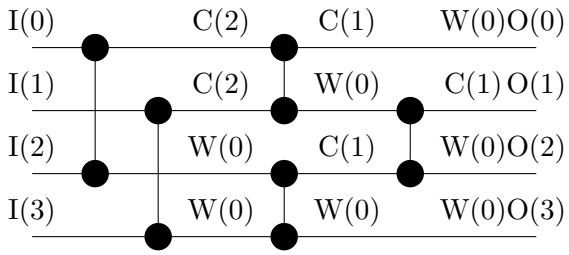
L-systém navržený pro tuto práci je parametrický přepisovací systém s **abecedou** $V = \{I, O, W, C\}$ a s parametry p, i a k. Význam jednotlivých symbolů je následující:

- $I(i)$ (*input*) je vstup řadicí sítě, kde i je číslo vodiče,
- $O(i)$ (*output*) je výstup řadicí sítě, kde i je číslo vodiče,
- $W(p)$ (*wire*) je drát v řadicí síti (bez komparátoru) a jeho parametr nemá význam při interpretaci,
- $C(p)$ (*input*) je komparátor v řadicí síti, který spojuje vodiče i a $i + p$,
- i je číslo vodiče, k je pořadové číslo derivace a p je proměnná zastupující obecný vyčíslitelný parametr (numerickou hodnotu, funkci, výraz).

Axiomem navrženého L-systému může být například prázdná síť reprezentovaná posloupností symbolů
$I(0)O(0)I(1)O(1)\dots I(n)O(n)$ pro všechna $0 \leq i < n$, kde n je počet vstupů sítě. Dále můžeme jako axiom využít například funkční optimální řadicí síť se 4 vodiči viz obrázek

5.1. Posloupnost symbolů pro tento axiom by byla následující:

$$\begin{aligned} &I(0)C(2)C(1)W(0)O(0)\\ &I(1)C(2)W(0)C(1)O(1)\\ &I(2)W(0)C(1)W(0)O(2)\\ &I(3)W(0)W(0)W(0)O(3) \end{aligned} \tag{5.1}$$



Obrázek 5.1: Axiom řadicí sítě (optimální čtyřvstupá síť)

Na každém řádku jsou symboly určující komponenty pro jeden vodič. Jedná se o jeden celistvý řetězec pro L-systém. Rozdělení na jednotlivé řádky je zavedeno pouze pro lepší přehlednost. V daném řádku je vždy nejprve symbol pro vstup, následují dráty a komparátory a řádek končí symbolem pro výstup. Počet komponent ve všech řádcích je shodný, přičemž tato vlastnost je důležitá pro interpretaci. Poloha komponenty v řádku určuje pozici dané komponenty ve struktuře řadicí sítě.

Pravidla tohoto L-systému jsou v obecném tvaru:

$$\begin{aligned} &O(i) : (condition) \rightarrow C(p)O(i) \quad nebo\\ &O(i) : (condition) \rightarrow W(p)O(i) \end{aligned} \tag{5.2}$$

Z podoby pravidel a axiomu vyplývá, že výstup v každé derivaci přepíšeme na komparátor nebo prázdný vodič, což nás posune do další vrstvy. K vložené komponentě je také zduplikován výstup, abychom mohli pokračovat v přepisování dalšími derivacemi. Tato varianta umožní přidat na každý vstup v jedné derivaci jednu komponentu. To ovšem nemusí být dostačující. Délka následníku pravidla proto může být i delší. Tuto velikost je možno změnit v konfiguračním souboru pro evoluční algoritmus `params.h`. Pravidla L-systému budou mít následující obecný tvar:

$$O(i) : (condition) \rightarrow [C(p)W(p)]^{+}O(i) \tag{5.3}$$

Při splnění podmínky by se výstup přepsal na předepsaný počet komponent C a W následovaný opět výstupem. Tato možnost nastavení je důležitá zejména proto, že dovoluje regulovat počet komponent přidaných do sítě v jedné derivaci. Přidáním počtu komponent tak můžeme snížit počet derivací potřebných k jedné etapě růstu.

Pokud není splněna podmínka žádného pravidla je aplikováno pravidlo univerzální. Toto pravidlo zajistí vložení potřebného počtu prázdných komponent $W(p)$, čímž zůstane zachována stejná šířka všech sekcí řetězce.

$$O(i) \rightarrow [W(p)]^{+}O(i) \tag{5.4}$$

Podmínka pravidla má obecnou podobu *a operator b porovnani c*. Parametr p užitý v komponentách má tvar matematického výrazu *d operator e*. Číselné konstanty a, b, c, d a e mohou reprezentovat velikost sítě, pořadové číslo derivace, číslo vstupu nebo číselnou hodnotu. Operátor *operator* je binární a může se jednat o násobení, dělení, sčítání, odčítání, modulo, minimum nebo maximum. Pro porovnání byly použity tyto operace: $<, \leq, >, \geq, =$. Pravidla L-systému včetně jejich podmínek a parametrů jsou navrhována EA.

Protože součástí podmínek a parametrů pravidel může být pořadové číslo derivace dochází k vyčíslení parametru p v komponentách již při samotné derivaci. Jednou vygenerované komponenty se tak již nemění.

5.1.1 Praktický příklad navrženého systému

Příkladem navrženého L-systému generujícího řadicí síť se čtyřmi vodiči je tento: $G = (V, \omega, P)$, kde: abeceda $V = \{I, O, W, C\}$, axiom $\omega = I(0)O(0)I(1)O(1)I(2)O(2)I(3)O(3)$ a množina pravidel P obsahuje šest pravidel, kde i je číslo vstupu a k pořadové číslo derivace:

$$\begin{aligned}
&1) \quad O(i) : (k - 2 = i) \rightarrow C(2-1)O(i) \\
&2) \quad O(i) : (i : k = 2) \rightarrow C(2-1)O(i) \\
&3) \quad O(i) : (k * i = 6) \rightarrow C(2-3)O(i) \\
&4) \quad O(i) : (k * i \geq 3) \rightarrow W(k-k)O(i) \\
&5) \quad O(i) : (i + 2 \geq k) \rightarrow C(2 * k)O(i) \\
&6) \quad O(i) \rightarrow W(0)O(i)
\end{aligned} \tag{5.5}$$

Nyní provedeme 3 derivace, jimiž vygenerujeme řetězec reprezentující funkční řadicí síť o čtyřech vstupech. Pro lepší orientaci v řetězci jsou v axiomu a derivacích odřádkovány komponenty pro jednotlivé vodiče. Průběh derivování je následující :

$$\begin{aligned}
\omega = I(0)O(0) \quad & aplikujeme\ pravidlo\ 5 \\
I(1)O(1) \quad & aplikujeme\ pravidlo\ 5 \\
I(2)O(2) \quad & aplikujeme\ pravidlo\ 2 \\
I(3)O(3) \quad & aplikujeme\ pravidlo\ 4
\end{aligned} \tag{5.6}$$

$$\downarrow \quad 1.derivace, k = 1$$

$$\begin{aligned}
I(0)C(2)O(0) \quad & aplikujeme\ pravidlo\ 1 \\
I(1)C(2)O(1) \quad & aplikujeme\ pravidlo\ 5 \\
I(2)C(1)O(2) \quad & aplikujeme\ pravidlo\ 4 \\
I(3)W(0)O(3) \quad & aplikujeme\ pravidlo\ 3
\end{aligned} \tag{5.7}$$

$$\downarrow \quad 2.derivace, k = 2$$

$$\downarrow \quad 2.derivace, k = 2$$

$$\begin{aligned}
&I(0)C(2)C(1)O(0) \quad aplikujeme\ pravidlo\ 6\\
&I(1)C(2)C(4)O(1) \quad aplikujeme\ pravidlo\ 1\\
&I(2)C(1)W(0)O(2) \quad aplikujeme\ pravidlo\ 3\\
&I(3)W(0)C(-1)O(3) \quad aplikujeme\ pravidlo\ 4
\end{aligned} \tag{5.8}$$

$$\downarrow \quad 3.derivace, k = 3$$

$$\begin{aligned}
&I(0)C(2)C(1)W(0)O(0)\\
&I(1)C(2)C(4)C(1)O(1)\\
&I(2)C(1)W(0)C(-1)O(2)\\
&I(3)W(0)C(-1)W(0)O(3)
\end{aligned} \tag{5.9}$$

5.2 Možnosti interpretace řetězců

Interpretace řetězců vygenerovaných navrženým L-systémem se může lišit v závislosti na použitém interpretu. Anylýza struktur řadicích sítí odhalila různé možnosti konstrukce řadicích sítí z řetězce vygenerovaného L-systémem. V této práci budou zkoumány dvě varianty interpretace.

První metoda pro konstrukci řadicí sítě využívá želví grafiku. Tuto metodu interpretace nazýváme „interpretace po vodičích“, protože želva postupuje popořadě po jednotlivých sekcích reprezentujících vodiče.

Druhá metoda vychází z konvenčních postupů tvorby řadicích sítí a je zde nazvána „interpretace po vrstvách“.

Podrobnému popisu těchto metod jsou věnovány samostatné kapitoly 5.3 a 5.4. V těchto kapitolách je vždy detailně vysvětlen princip dané metody a všechny techniky jsou demonstrovány na praktickém příkladu (převedení poslední derivace 5.9 na odpovídající řadicí síť).

5.3 Interpretace L-systému po vodičích

První typ interpretu, který bude představen, byl inspirován želví grafikou. V tomto případě želva čte popořadě jednotlivé znaky s jejich parametry v řetězci vygenerovaném L-systémem a interpretuje je.

Pomocí této metody bude experimentálně ověřen nekonvenční přístup k tvorbě řadicích sítí. Tato metoda bude v kapitole 7 srovnána s více konvenční metodou tvorby sítě „po vrstvách“.

Interpretace symbolů abecedy želvou je následující:

- $I(i)$ vytvoř vstup na vodiči i a dále se pohybuj po tomto vodiči,
- $O(i)$ vytvoř výstup vodiče i a přesuň se na začátek vodiče $i + 1$,

- W vlož prázdný vodič (posuň se o vrstvu dále),
- $C(p)$ spoj komparátorem vodič i (po kterém se pohybuješ) s vodičem $i+p$ (posuň se o vrstvu dále).

Při interpretaci symbolů W ani C se želva neposunuje z aktuálního vodiče na jiný. Při vkládání komparátorů také želva kontroluje, zda jsou vkládané komparátory validní (nezasahují mimo síť).

V průběhu interpretace jednotlivých vodičů může docházet ke kolizím komparátorů v jednotlivých vrstvách. Aby se komparátory mohly nacházet v jedné vrstvě nesmí mít žádné společné vstupy (výstupy).Nabízí se tedy otázka co udělat s komparátory, které do „své“ vrstvy nemohou být zařazeny. První možností je kolidující komparátory úplně vyloučit z konstrukce řadicí sítě. Tato možnost však odstraní poměrně velké množství komparátorů, což se ve výsledku může projevit primárně na funkčnosti sítě sekundárně potom na době běhu EA. Druhou možností je naopak nevyhovující komparátory ponechat, pouze je posunout do vrstvy, ve které nebudou kolidovat s jiným komparátorem. V následujících podsekcích je popsán princip interpretace pro obě tyto naimplementované varianty.

Vzhledem k podobě řetězce a jeho interpretaci zleva doprava je jasné, že nejprve budou vloženy všechny komponenty nacházející se na prvním vstupu. Toto posléze může omezit komponenty vycházející z dalších vrstev, kterým budou v připojení na nižší vstupy bránit již existující komparátory. Z podoby insertion a bubble sortu viz obrázek 4.7 na straně 18, které také používají více komponent na vstupech s nižšími indexy, lze usoudit, že toto omezení nemusí být příliš škodlivé.

5.3.1 Odstranění kolidujících komparátorů

Interpretace symbolů pro želvu zůstává téměř beze změny. Jedinou změnou je rozšíření omezení při vložení komparátoru. Pokud se želva dostane k interpretaci symbolu $C(p)$, nejprve zkontroluje, zda jsou vstupy vkládaného komparátoru v rozsahu sítě. Pokud je komparátor takto validní, dochází k sekundární kontrole, zda vstupy komparátoru nekolidují s jiným komparátorem v téže vrstvě. Pokud splňuje komparátor obě omezení, je vložen. Pokud nikoli, je na jeho místo vložen prázdný drát, želva komparátor zahodí a přesune se do další vrstvy k interpretaci následujícího znaku.

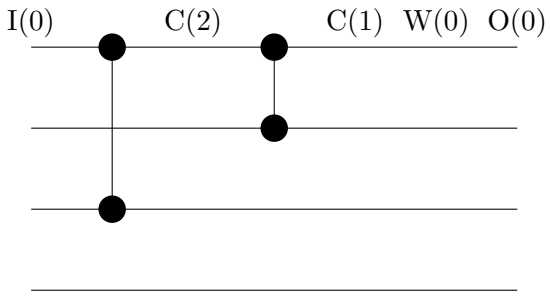
Vlastností této interpretace je, že poloha komponenty v řetězci přímo určuje vrstvu, ve které se komponenta nachází. Počet komponent v jedné sekci K pak tedy určuje výsledné zpoždění řadicí sítě *zpoždění*$= K-2$ (komponenty vstupu a výstupu nezpůsobují zpoždění).

Na následujících obrázcích můžete vidět postupnou interpretaci řetězce L-systému představeného v sekci 5.1.1 na straně 21. Jedná se o jednoduchou čtyřvstupou síť, které vznikla po provedení 3. derivace tohoto přepisovacího systému. Viz řetězec 5.10 (totožné s řetězcem 5.9 na straně 22).

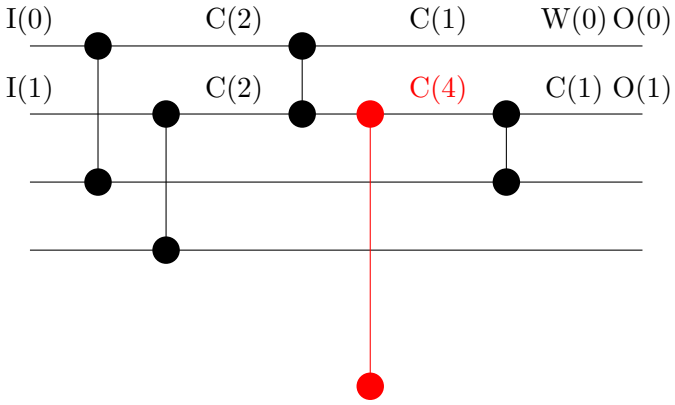
$$\begin{aligned}
&I(0)C(2)C(1)W(0)O(0)\\
&I(1)C(2)C(4)C(1)O(1)\\
&I(2)C(1)W(0)C(-1)O(2)\\
&I(3)W(0)C(-1)W(0)O(3)
\end{aligned} \tag{5.10}$$

Ačkoli jsou vizuálně odděleny sekce představující komponenty pro jednotlivé vodiče, řetězec je vnímán jako souvislý. Je interpretován zleva znak po znaku. Červeně jsou v řetězci vyznačeny komparátory, které budou z důvodu nesplnění podmínek zahozeny.

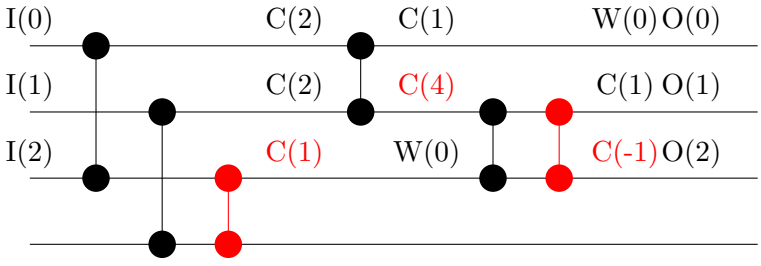
Pro první řádek jsou všechny komponenty vloženy beze změny (obr. 5.2). Pro druhý řádek je odstraněn červeně vykreslený komparátor $C(4)$ (obr. 5.3). Jedná se o komparátor z aktuálního vodiče 2 na vstup 6 zasahující mimo síť. Červeně vykreslené komparátory jsou z výsledné sítě odstraněny. Pro třetí vodič je odstraněn červený komparátor $C(1)$, který by kolidoval s komparátorem $(C2)$ z prvního vodiče. Také je odstraněn červený komparátor $C(-1)$, který koliduje s komparátorem $C(1)$ z druhého vodiče (obr. 5.4). Na posledním vodiči jsou opět vloženy všechny komponenty beze změny (obr. 5.5).



Obrázek 5.2: Interpretace komponent pro 1. vodič, vložení všech komponent

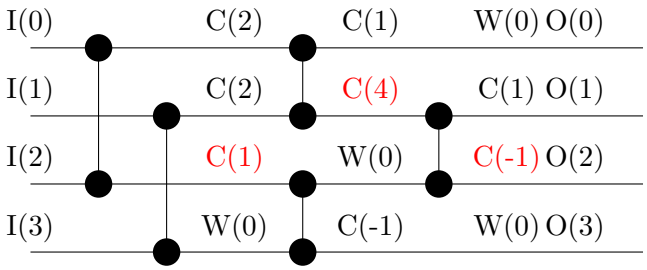


Obrázek 5.3: Interpretace komponent pro 2. vodič, zahození červeného komparátoru $C(4)$



Obrázek 5.4: Interpretace komponent pro 3. vodič, zahození dvou červených komparátorů $C(1)$ a $(C-1)$

Z obrázků je zřejmé, že se podařilo sestrojit funkční čtyřvstupou síť. Nicméně i na takto malé síti došlo kvůli kolizím komponent k odstranění 2 komparátorů z původních 8, což



Obrázek 5.5: Interpretace komponent pro 4. vodič, beze změny komponent

je čtvrtina. V tomto případě komparátory byly zbytečné avšak u jiných sítí můžeme přijít o potřebné komponenty.

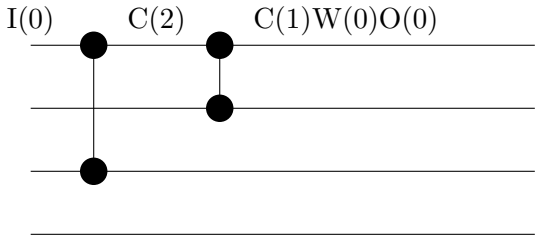
5.3.2 Posun kolidujících komparátorů

Pokud chceme zachovat všechny komparátory vygenerované L-systémem, je třeba najít způsob, jak je zařadit do řadicí sítě tak, aby nebyly v kolizi s ostatními. Nejjednodušším způsobem, jak toho dosáhnout, je komparátor posunout do další vrstvy. Pokud jsou v další vrstvě volné požadované vstupy, je komparátor umístěn. Pokud se objeví kolize i v další vrstvě, posunuje se komparátor, dokud není nalezena vhodná vrstva.

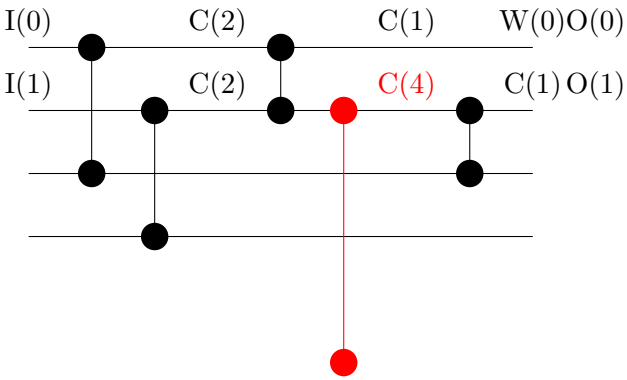
Pro názornou ukázku bude použit stejný předpis pro řadicí síť jako v předchozím případě (řetězec s vyznačenými kolizemi viz 5.11). Jediným rozdílem bude to, že komparátory $C(1)$ a $C(-1)$ ze třetího vodiče budou zařazeny do jiných vrstev. Komparátor z druhého vodiče $C(4)$ bude zahozen. Zasahuje mimo síť, takže ani posunem do další vrstvy by ho nebylo možno umístit.

$$
\begin{aligned}
&I(0)C(2)C(1)W(0)O(0)\\
&I(1)C(2)C(4)C(1)O(1)\\
&I(2)C(1)W(0)C(-1)O(2)\\
&I(3)W(0)C(-1)W(0)O(3)
\end{aligned}
\tag{5.11}
$$

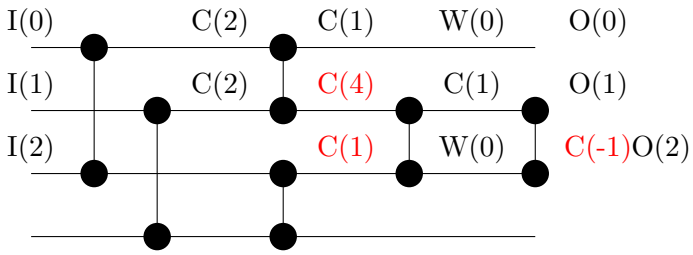
Pro první řádek jsou opět vloženy všechny komponenty (obr. 5.6). Na druhém řádku se zahodí komparátor $C(4)$ zasahující mimo síť (obr. 5.7). Na třetím vstupu dojde k posunu komparátoru $C(1)$ o jednu vrstvu. Posunutím tohoto komparátoru se komparátor C(-1) na stejné vstupu dostane také o vrstvu dále a zde již nekoliduje (obr. 5.8). Na posledním vstupu se díky předchozím posunů dostal komparátor $C(-1)$ taktéž do kolize a je třeba jej posunout ze 3. až do 6. vrstvy (obr. 5.9).



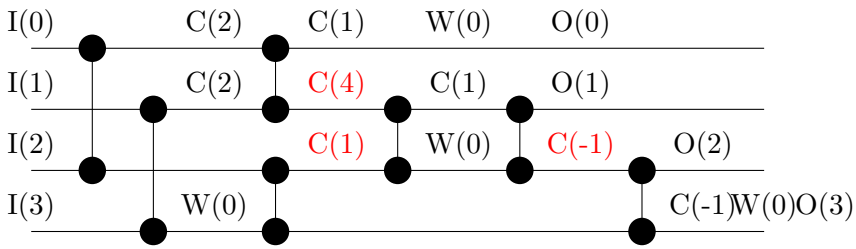
Obrázek 5.6: Interpretace komponent pro 1. vstup, žádná kolize, totožné s předchozím přístupem



Obrázek 5.7: Interpretace komponent pro 2. vstup, zahození červeného komparátoru $C(4)$ (mimo síť)



Obrázek 5.8: Interpretace komponent pro 3. vstup, posun červeného komparátoru $C(1)$ o 1 vrstvu



Obrázek 5.9: Interpretace komponent pro 4. vstup, posun kolidujícího komparátoru $C(-1)$ o 2 vrstvy

5.4 Interpretace L-systému po vrstvách

Další metoda konstrukce řadicích sítí z řetězce L-systému je inspirována konvenčními postupy tvorby řadicích sítí. Řadicí sítě jsou tvořeny postupně po jednotlivých vrstvách. Na L-systému ani výpočtech se nic nemění. Jedinou změnou je úprava interpretace.

Tuto interpretaci již není možno považovat za želví grafiku. Interpretace totiž neprobíhá znak po znaku tak, jak jdou za sebou. Vygenerovaný řetězec si rozdělíme na jednotlivé sekce pro každý vstup. Každému znaku C a W je přidělen index popisující jeho pozici uvnitř sekce. Poté dochází k postupnému zpracovávání všech znaků. V jednom kroku jsou vždy zpracovávány znaky ze všech sekcí, které mají stejný index (jedna vrstva sítě).

Opět nemůžeme předpokládat pouze ideální průběh interpretace bez kolidujících komparátorů v jedné vrstvě. I pro tento případ tedy existují dvě nastavení interpretu. První nastavení kolidující komparátory zahazuje, druhé nastavení je odsouvá do dalších vrstev.

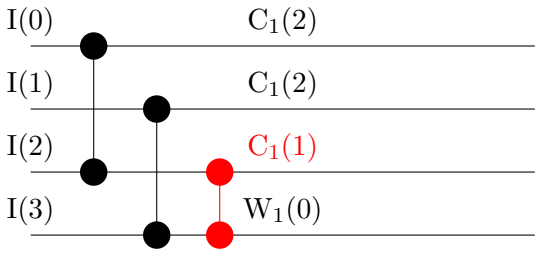
5.4.1 Odstranění kolidujících komparátorů

V průběhu interpretace jednotlivých vrstev jsou nejprve vkládány komparátory z nižších vstupů. Při vkládání opět probíhá kontrola, zda komparátor nezasahuje mimo síť a zda nekoliduje s jiným v aktuální vrstvě. Při nesplnění omezení je komparátor zahozen.

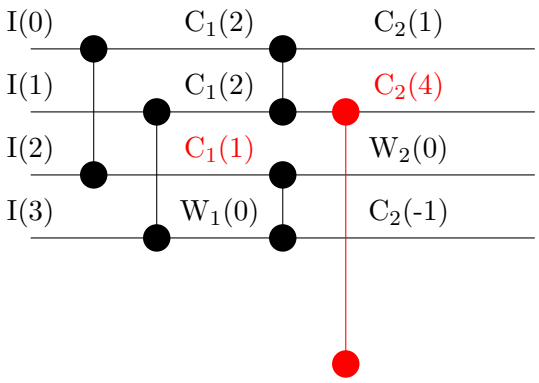
Pro demonstraci tohoto přístupu byl použit opět stejný řetězec (tři derivace L-systému představeného v sekci 5.1.1 na straně 21). Zahozené komparátory jsou opět označeny červeně a jednotlivé komponenty mají přiřazeny odpovídající indexy.

$$
\begin{aligned}
&I(0)C_1(2)C_2(1)W_3(0)O(0)\\
&I(1)C_1(2)C_2(4)C_3(1)O(1)\\
&I(2)C_1(1)W_2(0)C_3(-1)O(2)\\
&I(3)W_1(0)C_2(-1)W_3(0)O(3)
\end{aligned}
\tag{5.12}
$$

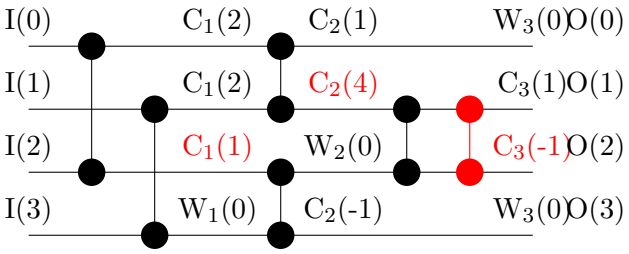
Na následujících obrázcích jsou zobrazeny jednotlivé kroky vývoje sítě. Pro index 1 je zahozen kolidující komparátor $C_1(1)$ na třetím vstupu (obr.5.10). Pro index 2 je zahozen komparátor $C_2(4)$ zasahující mimo síť (obr. 5.11). Při zpracování indexu 3 je zahozen kolidující komparátor $C_3(-1)$ (obr. 5.12).



Obrázek 5.10: Interpretace komponent pro index 1, odstranění kolidujícího komparátoru $C_1(1)$



Obrázek 5.11: Interpretace komponent pro index 2, odstranění komparátoru $C_2(4)$ (mimo síť)

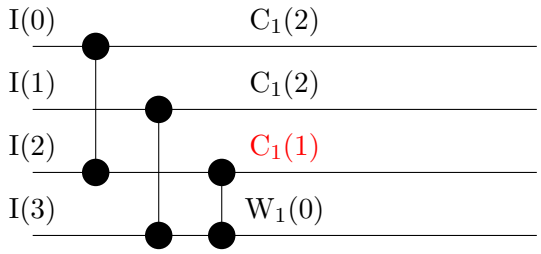


Obrázek 5.12: Interpretace komponent pro index 3, odstranění kolidujícího komparátoru $C_3(-1)$

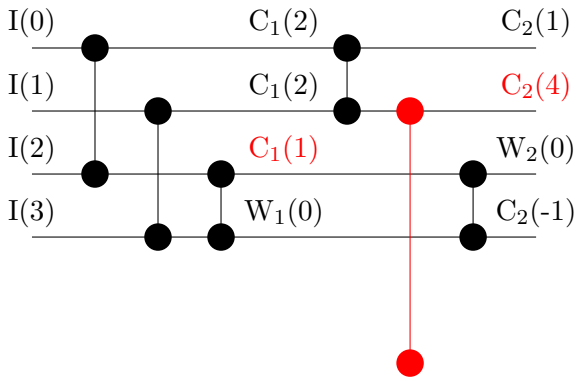
5.4.2 Posun kolidujících komparátorů

Předchozím přístupem můžeme ztratit některé z potřebných komparátorů. Řešením je tedy opět odsun komparátoru do dalších vrstev. Protože teď však probíhá interpretace po vrstvách, mají přednost komparátory s nižším indexem a nedochází k tak rozsáhlým posunům. Při interpretaci po vodičích se totiž může snadno stát, že komparátory z posledních vstupů jsou odsunuty až do úplně posledních vrstev řadicí sítě. Při tvorbě po vrstvách se komparátor přesune pouze do vedlejší, zatím ještě prázdné, vrstvy. V programu je toto implementováno zjednodušeně tak, že každý komparátor má svou vrstvu. Na výsledný výpočet zpoždění toto zjednodušení nemá vliv, viz obrázky 5.15 a 5.16.

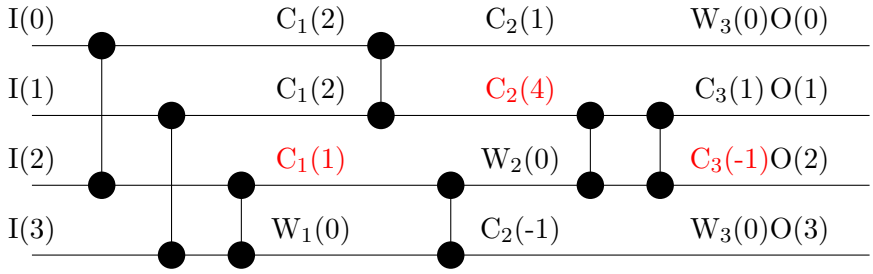
Na následujících obrázcích je demonstrována interpretace tvorby po vrstvách s posunutím. K předvedení metody je užit již známý řetězec třetí derivace L-systému představeného v sekci 5.1.1 na straně 21.



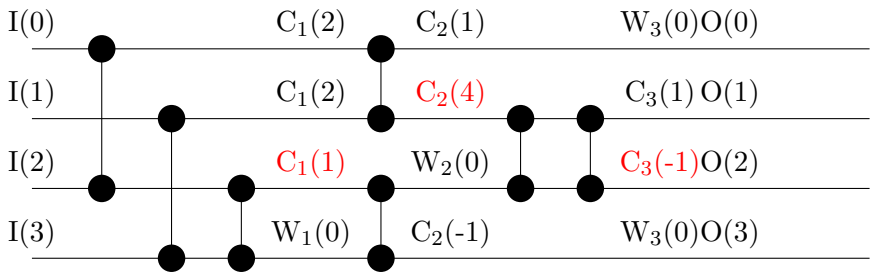
Obrázek 5.13: Interpretace komponent pro index 1, posunutí kolidujícího komparátoru $C_1(1)$



Obrázek 5.14: Interpretace komponent pro index 2, odstranění komparátoru $C_2(4)$ (mimo síť)



Obrázek 5.15: Interpretace komponent pro index 3, posunutí kolidujícího komparátoru $C_3(-1)$



Obrázek 5.16: Stejná síť jako na obrázku 5.15 zakreslena paralelně, zpoždění = 5

Kapitola 6

Implementovaný evoluční návrh L-systému

Výše popsaný systém byl použit pro samotnou implementaci EA a jeho jednotlivých variant v mé práci. Evolučně je navrhován předem daný počet pravidel L-systému včetně jejich podmínek a parametrů. Počet komponent v následníku je předem dán uživatelem.

Každý chromosom je reprezentován strukturou ve které se nachází fitness hodnota, pravidla, (optimalizovaný) počet komparátorů a zpoždění pro poslední interpretaci a parametr zda byl jedinec už vyhodnocen.

Struktura pravidla obsahuje pole integerů reprezentujících podmínku, pole s komponentami následníka a parametr zda bylo toto pravidlo použito při derivacích L-systému.

Samotná implementace pro návrh L-systému generujícího rostoucí řadicí sítě je rozdělena do několika částí, přičemž každá má na starosti určitou část evolučního algoritmu.

6.1 Hlavní část

Hlavní část programu v souboru `main.cpp` se stará o běh EA, volá jednotlivé funkce EA ve správném pořadí a kontroluje zda nebyl nalezen v aktuální generaci výsledek. Hlavní část programu také inicializuje generátor pseudonáhodných čísel, načítá vstupní soubory a tiskne na výstup případné výsledky.

6.2 Části evolučního algoritmu

Tato část programu v souboru `eaparts.cpp` obsahuje všechny funkce důležité pro běh EA. Můžete zde nalézt například výpočet fitness funkce, křížení, mutace, či klasický turnajový výběr se dvěma náhodně vybranými soutěžícími. Dále jsou zde implementovány podpůrné funkce pro evaluaci jedinců a pomocné funkce pro tisk.

6.2.1 Fitness funkce

Výpočet fitness je jedním z nejdůležitějších v celém evolučním algoritmu. Její hodnota reprezentuje přesnost sestrojené řadicí sítě, tedy počet správně seřazených bitů pro všechny možné posloupnosti nul a jedniček. Například při velikosti řídí sítě $pocet\_vstupu = 4$ musí být správně seřazeno 2^4 vstupních posloupností, přičemž každá tato posloupnost má 4 bity.

Maximální hodnota fitness funkce pro tento případ by tedy byla $2^4 * 4 = 64$. Obecný vzorec pro výpočet maximální fitness funkce pro danou síť je následující:

$$max\_fitness = 2^{pocet\_vstupu} * pocet\_vstupu \tag{6.1}$$

Pokud chceme evolučním návrhem objevit sítě rostoucí, je potřeba vyhodnotit všechny řadicí sítě při jejich postupném růstu pro daný počet „růstových kroků“. Při každém vyhodnocení fitness funkce je tedy proveden požadovaný počet derivací. V derivacích určených uživatelem je vyhodnocena aktuální řadicí síť a jsou přidány další vodiče k již existující síti. Maximální hodnota fitness funkce je v tomto případě rovna součtu maxim pro jednotlivé dílčí sítě. Příkladem může být 16-vstupá síť, rostoucí z funkčního 4-vstupého axiomu. Evoluční algoritmus má za úkol najít takový L-systém, který vždy po dvou derivacích vygeneruje funkční síť větší o 4 vodiče než byla síť předchozí. Postup bude následující:

1. Vynulování celkové fitness jedince.
2. Doplnění axiomu/existující sítě o 4 vodiče.
3. Provedení dvou derivací L-systému.
4. Vyhodnocení aktuální sítě a připočtení tohoto výsledku k celkové fitness jedince.
5. Pokud jsme dosáhli maximální velikosti sítě výpočet končí, pokud ne návrat zpět k bodu 2.

Obecný předpis pro maximální hodnotu fitness funkce pro rostoucí síť má tvar:

$$max\_fitness = \sum 2^v * v; v = a + x * s \wedge v \leq v_{max} \wedge x \in \mathbb{N}, \tag{6.2}$$

kde v je počet vstupů sítě, a je počet vstupů axiomu, s je počet vstupů o který roste síť v jednom kroku a v_{max} je počet vstupů finální sítě.

Pro optimalizaci sítě by bylo možné zavést do výpočtu fitness funkce penalizace za velké zpoždění či vysoký počet komparátorů. Tato úprava by však narušila povrch fitness funkce a kvůli penalizaci bychom mohli přijít o zajímavé výsledky. Vzhledem k tomu, že cíl práce je najít především rostoucí sítě, je optimálnost sítí až druhořadým problémem. U nalezených výsledků však dochází k optimalizaci odstraněním redundantních komparátorů.

6.2.2 Křížení, mutace a selekce

Další nedílnou součástí evolučního algoritmu jsou operátory křížení, mutace a selekce. Mutace je implementována klasicky. U mutovaného jedince se vybere náhodné místo v genomu a jeden jeho gen je náhodně zmutován na jednu z přípustných hodnot.

Dále jsou implementovány dva typy křížení. Křížení jednobodové se dvěma rodiči, kdy se náhodně zvolí bod křížení na úrovni celých pravidel. Dále křížení pravidel v jednom jedinci, kdy se náhodně prohodí pořadí těchto dvou pravidel. Křížení pravidel v jednom jedinci je zavedeno z důvodu, že pravidla s nižším indexem mají při aplikaci na řetězec přednost. Vždy je aplikováno první pravidlo, jehož podmínka je vyhovující.

I drobnou úpravou L-systému můžeme zcela narušit dosavadní řešení. Z tohoto důvodu klasické jednobodové křížení není příliš vhodným evolučním operátorem a v mých finálních experimentech nebylo použito. Odkomentováním kódu jej ale může uživatel opět zařadit do výpočtu.

Jako operátor selekce je implementován klasický turnajový výběr. Jsou vybráni 2 jedinci z nichž lepší vítězí.

Do další generace vždy postupuje nejlepší jedinec, jeho mutant (pravděpodobnost mutace 100%) a jedinec vybraný turnajovým výběrem. U jedince vybraného turnajovým výběrem dochází spravděpodobností 80% ke řížení ve smyslu záměny pořadí pravidel.

6.3 Interpret

Interpret je v podstatě jádro celého algoritmu. Právě zde se rozhoduje jak bude prozatím „bezvýznamný“ řetězec L-systému převeden na výslednou řadicí síť. V této sekci programu se nacházejí funkce, které mají za úkol převod řetězce na odpovídající posloupnost komparátorů. Interpretace je možná po ukončení libovolné derivace.

Obecný význam jednotlivých symbolů i s jejich parametry byl již popsán v sekci 5.1. Právě v interpretaci řetězce na komparátory se liší jednotlivé varianty mnou testovaných evolučních algoritmů, proto jim byly věnované samostatné kapitoly 5.3 a 5.4.

6.4 Zero-one

Tato část programu dostane na vstup posloupnost komparátorů pro daný počet vstupů, kterou vygeneroval interpret. Jsou vygenerovány testovací sekvence, na které jsou postupně aplikovány jednotlivé komparátory. Po aplikaci celé posloupnosti komparátorů je ověřeno, kolik bitů testovacích v sekvencích je zařazeno na správném místě.

Při tomto ověřovaní také dochází k detekci komparátorů, které v žádné z testovaných sekvencí nezapříčinily výměnu hodnot na svých vstupech. Tyto komparátory jsou pro výslednou síť nadbytečné a jejich odstraněním snížíme jak zpoždění tak počet komponent.

Pro urychlení vyhodnocování byl použit paralelní přístup. Tento přístup využívá vlastnost principu zero-one, že při řazení posloupností složených pouze z nul a jedniček může komparátor fungovat jako bitový operátor AND a OR. Na výstupu komparátoru v_{min} bude výsledek operace OR a na výstupu v_{max} se bude nacházet výsledek operace AND. Takto je možno paralelně seřadit pomocí několika 32-bitových integerů 32 sekvencí současně. Implementace tohoto přístupu je inspirována vyhodnocováním algoritmu *EvoSort* vytvořeného M. Bidlem [4] a představuje značné urychlení výpočtu fitness funkce.

6.5 Konfigurační soubor EA

Součástí programu je také konfigurační soubor `params.h`, ve kterém uživatel může nastavit parametry EA jako např. pravděpodobnosti mutace a křížení či maximální délku běhu programu v generacích.

Také zde lze nastavit parametry hledaného L-systému systému a to axiom, délku následníka pravidla, výslednou velikost sítě, velikost kroku růstu a požadovaný počet derivací mezi jednotlivými kroky.

Po každé změně parametrů je potřeba celý program opět přeložit a znovu spustit.

Kapitola 7

Experimentální výsledky

V této kapitole jsou shrnuty výsledky experimentů pro jednotlivé varianty interpretace. Bude ukázáno, že představené techniky jsou schopny navrhnout L-systémy generující řadicí sítě požadované velikosti. Některé takto navržené systémy jsou schopny v dalších derivacích generovat větší řadicí sítě. Takto navržené L-systémy tedy reprezentují (částečně) rostoucí sítě.

Nejlepším výsledkem experimentů je L-systém navržený evolucí pro růst do 24 vstupů z čtyřvstupého axiomu. V každém kroku růstu je provedena 1 derivace a jsou přidány 4 vodiče. Systém obsahuje 5 pravidel, z nichž každé má v následníku 5 komponent (C nebo W). Bylo ověřeno, že tento L-systém navržený evolucí je schopen v následujících třech derivacích vygenerovat síť až o 36 vstupech.

Pro každou variantu interpretu budou uvažována nastavení z tabulky 7.1. Jednotlivá nastavení A-E byla testována s různým počtem pravidel (5, 12) a dvěma různými axiomy (prázdná síť se 4 vodiči - 4P, funkční síť se 4 vodiči - 4F). Celkem bylo vyzkoušeno 20 různých nastavení pro každou variantu interpretu. Nastavení bude dále označováno řetězcem $nastaveni\_pocetPravidel\_typAxiomu$. Například nastavení A s pěti pravidly L-systému a čtyřvstupým axiomem reprezentujícím funkční řadicí síť by bylo zapsáno jako A_5_4F.

Z testování v průběhu vývoje aplikace dopadly nejlépe testy s pravděpodobností mutace 100% a křížení pořadí pravidel 80%. Křížení pravidel dvou jedinců se ukázalo jako nevhodné z důvodu přílišného narušování částečných řešení. Tyto hodnoty pravděpodobností byly použity pro finální testování.

Pro každé nastavení bylo spuštěno cca 100 nezávislých běhů. U evolučně navržených L-systémů byl ověřen růst sítí v dalších derivacích do velikosti 36 vstupů. Úspěšnost a nejlepší výsledky jednotlivých variant interpretu jsou představeny v následujících samostatných podkapitolách.

nastavení	A	B	C	D	E
počet vstupů na krok růstu	2	3	3	4	4
počet derivací na krok růstu	1	2	1	2	1
počet komponent v pravé straně pravidla	2	2	3	3	5
výsledná velikost při evoluci	22	19	19	24	24

Tabulka 7.1: Jednotlivá nastavení EA pro finální testování

7.1 Interpretace po vodičích

Pomocí této metody interpretace je možno evolučně navrhnout řadicí sítě, které rostou do zadané velikosti. V dalších derivacích L-systému některé z takto navržených sítí generují funkční sítě s více vodiči. Celková průměrná úspěšnost evoluce pro všechna nastavení je 9%. Tato hodnota je však zkreslena tím, že úspěšnost interpretu s odstraněním komparátorů je menší než 1%.

7.1.1 Odstranění komparátorů

Úspěšnost evoluce pro tuto variantu interpretace je menší něž 1%. Z finální sady výsledků vzešly jen 2 výsledky z nichž ani jeden nebyl rostoucí.

Předpoklad, že tato metoda odstraní velké množství komparátorů, které budou ve výsledné síti chybět, se ukázal jako pravdivý.

Pro tuto metodu byly spuštěny doplňující experimenty, ve kterých byl navýšen počet derivací na krok růstu a počet komponent v následníku pravidla. V této doplňující sadě vzrostla úspěšnost evoluce na 2%. Bohužel ani žádný z takto navržených systémů nebyl schopný generovat větší sítě, než pro které byl navržen evolucí.

Jedinou výhodou metody odstranění komparátorů je generování sítí, které jsou oproti metodě posunu komparátorů optimálnější jak počtem komparátorů tak i zpožděním. Počet odstraněných redundantních komparátorů je tak mnohem nižší.

Nejlepší výsledky

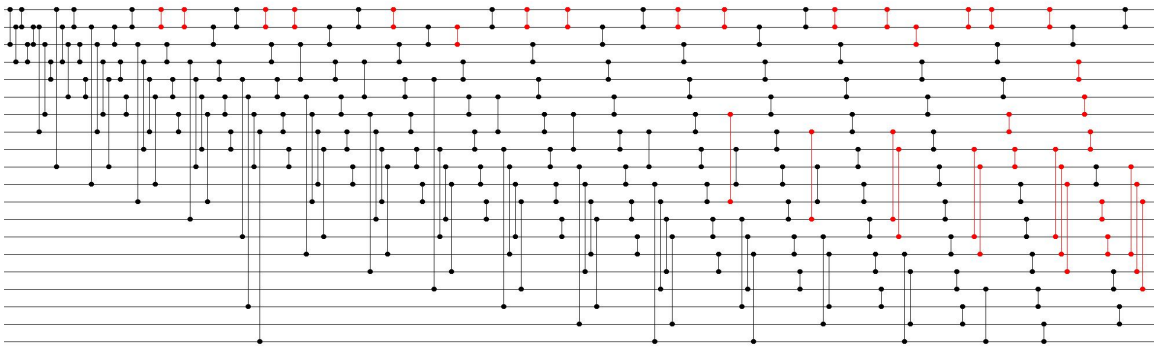
Výsledkem s nejzajímavější strukturou řadicí sítě je L-systém z doplňujících experimentů. Tento systém byl navržen pro růst do 20 vstupů a dále už neroste. Jeho vývoj začínal z axiomu reprezentujícího čtyřvstupou funkční řadicí síť. V jednou růstovém kroku jsou provedeny 2 derivace a jsou přidány 2 vodiče. Počet pravidel je 12 a v následníku pravidla jsou 2 komponenty.

Navržený L-systém je tento: $G = (V, \omega, P)$, kde abeceda $V = \{I, O, C, W\}$, axiom $\omega = I(0)C(2)C(1)W(0)O(0)I(1)C(2)W(0)C(1)O(1)I(2)W(0)C(1)W(0)O(2)I(3)W(0)W(0)W(0)O(3)$ a množina pravidel P obsahuje 12 pravidel:

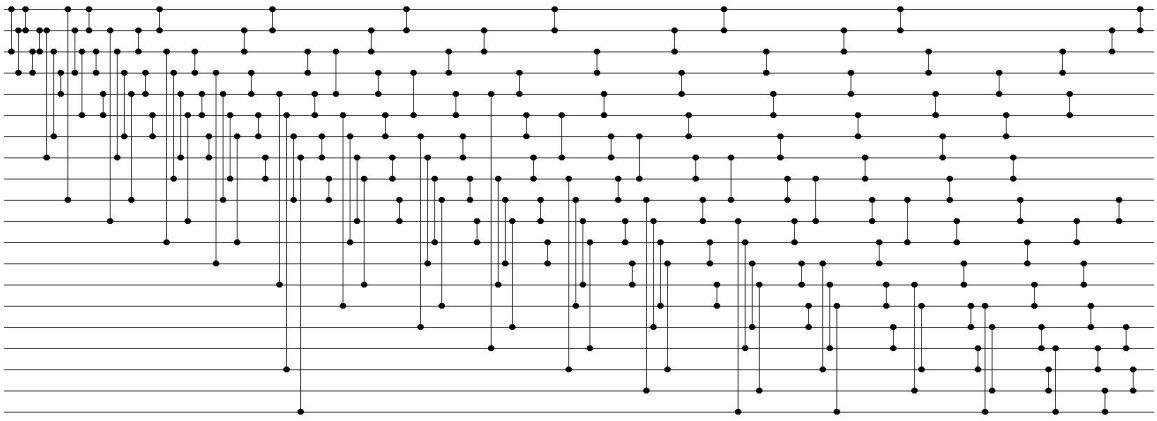
$$
\begin{aligned}
1)\quad & O(i) : (i - k < 1) \rightarrow C(2 * 0)C(5 + 0)W(0 * 1)O(i) \\
2)\quad & O(i) : (i * i = 5) \rightarrow C(1 * 0)C(5 * 0)W(0 * 1)O(i) \\
3)\quad & O(i) : (k - i < 1) \rightarrow C(5 - 0)C(2 - 1)W(0 * 1)O(i) \\
4)\quad & O(i) : (i - k = 2) \rightarrow C(1 + 0)C(5 - 0)W(0 * 1)O(i) \\
5)\quad & O(i) : (k\ max\ i < 2) \rightarrow C(3 * 0)C(2\%0)W(0 * 1)O(i) \\
6)\quad & O(i) : (k * i < k) \rightarrow C(k\%0)C(4/1)W(0 * 1)O(i) \\
7)\quad & O(i) : (k\ min\ k < 3) \rightarrow C(4\%0)C(3 + 1)W(0 * 1)O(i) \\
8)\quad & O(i) : (k/k = k) \rightarrow C(2 * 0)C(1\%0)W(0 * 1)O(i) \\
9)\quad & O(i) : (i\ min\ i < 3) \rightarrow C(5/0)W(k + 0)W(0 * 1)O(i) \\
10)\quad & O(i) : (k\%i < 5) \rightarrow C(0 + 0)C(3 * 0)W(0 * 1)O(i) \\
11)\quad & O(i) : (i\ min\ i > 4) \rightarrow C(1\%0)C(5 + 1)W(0 * 0)O(i) \\
12)\quad & O(i) : (k + k \neq 5) \rightarrow W(2 * 0)W(4 - 0W(0 * 0)O(i)
\end{aligned}
\tag{7.1}
$$

V poslední derivaci zobrazené na obrázku 7.1 má dvacetivstupá síť 196 komparátorů (161 po optimalizaci) a zpoždění 37 (35 po optimalizaci). Na obrázku 7.2 je zobrazena síť

po optimalizaci V obrázcích vizualizujících řadicí sítě neoptimalizované (pře odstraněním redundantních komparátorů) jsou redundantní komparátory zobrazeny červeně.



Obrázek 7.1: 8. derivace L-systému, funkční 20-vstupá řadicí síť před optimalizací



Obrázek 7.2: 8. derivace L-systému, funkční 20-vstupá řadicí síť po optimalizaci

7.1.2 Posun komparátorů

Úspěšnost evoluce pro tuto variantu interpretace je 18%. Z finální sady experimentů byli nejúspěšnější nastavení B (48%) a D (30%).

Ukázalo se, že různý počet pravidel nemá na procentuální úspěšnost ani schopnost růstu vliv. Pro oba počty pravidel se úspěšnost lišila maximálně o 5%. Pro menší počty pravidel jsou totiž při derivacích obvykle použita všechna pravidla. U vyššího počtu pravidel však roste počet pravidel, která nebyla aplikována v žádném derivačním kroku.

Evolučně úspěšnější byla nastavení využívající axiom s funkční řadicí sítí (23%). Nastavení využívající axiom s prázdnou čtyřvstupou sítí dosahovala průměrné úspěšnosti evoluce pouze 13%.

Při ověřování růstu sítí v dalších derivacích je úspěšnost pro různé počty pravidel a různé axiomy srovnatelná. O jednu derivaci je schopno vyrůst 13% z nalezených řešení, o dvě derivace je schopno vyrůst 6% nalezených řešení. Vyšší pravděpodobnost růstu mají zpravidla experimenty, u kterých je nastaven vyšší počet derivací na jeden růstový krok nebo vyšší počet komponent v následníku pravidla.

velikost sítě	19	22	25	28	21
počet komparátorů	264	351	451	562	686
počet komparátorů po optimalizaci	171	231	300	377	464
zpoždění sítě	56	65	75	86	96
zpoždění sítě po optimalizaci	37	45	51	60	65

Tabulka 7.2: Počet komparátorů a zpoždění v testovaných derivacích L-systému s nastavením B_5_4P

Nejlepší výsledky

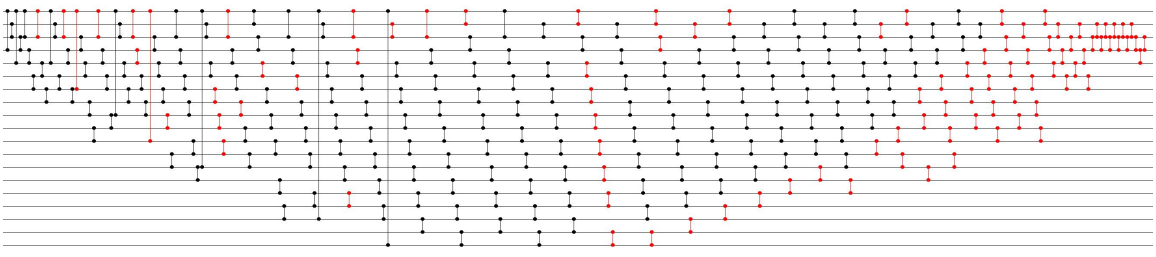
Pro nastavení B_5_4P byl navržen L-systém pro konstrukci sítě s 19 vodiči. V jednom růstovém kroku byly vždy provedeny 2 derivace a přidány 3 vodiče. Následník pravidel obsahovala 3 komponenty (C nebo W). Při následném ověření růstu tento systém prokázal schopnost generovat funkční sítě i v dalších 4 derivacích. Maximální velikost sítě generovaná tímto systémem je 31 vstupů. Síť generovaná v další derivaci již nebyla plně funkční.

Síť navržená tímto L-systémem využívá především krátké komparátory o délce 1. Vždy při růstovém kroku vkládá jeden komparátor přes celou síť.

V následující tabulce 7.2 je zobrazen počet komparátorů a zpoždění řadicích sítí vygenerovaných v jednotlivých derivacích pro nastavení B_5_4P. Na obrázku 7.3 je zobrazena 24-vstupá síť navržená evolucí před optimalizací. Na obrázku 7.4 je zobrazena optimalizovaná 24-vstupá síť navržená evolucí. V přílohách je na obrázku C.1 zobrazena optimalizovaná poslední funkční derivace navržená tímto systémem.

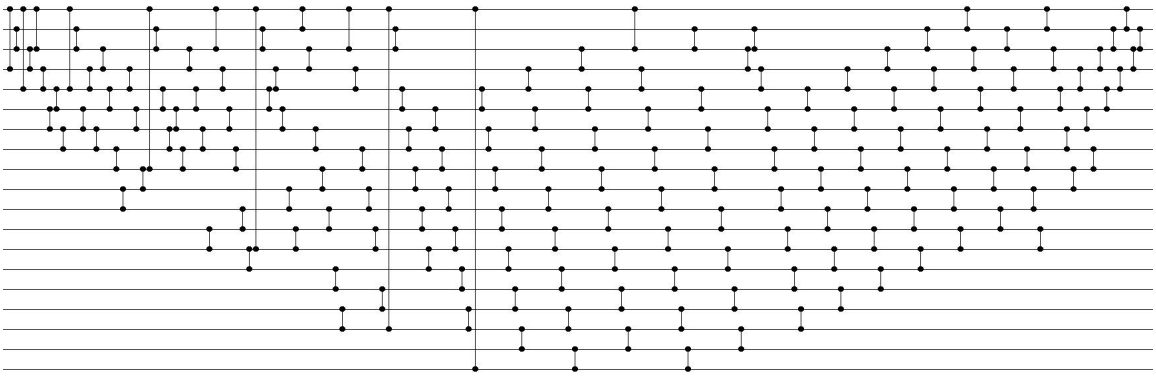
Navržený L-systém je tento: $G = (V, \omega, P)$, kde abeceda $V = \{I, O, C, W\}$, axiom $\omega = I(0)O(0)I(1)O(1)I(2)O(2)I(3)O(3)$ a množina pravidel P obsahuje 5 pravidel:

$$\begin{aligned}
&1) \quad O(i) : (i/i < 5) \rightarrow C(5\%1)C(3-1)O(i) \\
&2) \quad O(i) : (i/k < 5) \rightarrow C(k*1)C(2\ min\ 0)O(i) \\
&3) \quad O(i) : (i-k > 1) \rightarrow W(k/0)C(k/1)O(i) \\
&4) \quad O(i) : (k*i < k) \rightarrow C(3-1)C(0\ max\ 1)O(i) \\
&5) \quad O(i) : (i-k < 0) \rightarrow C(3+0)C(1\%0)O(i)
\end{aligned} \tag{7.2}$$



Obrázek 7.3: 10. derivace L-systému s nastavením B_5_4P, funkční 19-vstupá řadicí síť před optimalizací

Téměř totožnou síť generuje L-systém navržený pro nastavení B_12_4P. Tyto dva L-systémy mají kromě počtu pravidel stejné parametry a dorůstají do stejné velikosti.



Obrázek 7.4: 10. derivace L-systému s nastavením B_5_4P, funkční 19-vstupá řadicí síť po optimalizaci

velikost sítě	24	28	32	36
počet komparátorů	389	522	676	850
počet komparátorů po optimalizaci	231	311	407	795
zpoždění sítě	68	82	91	103
zpoždění sítě po optimalizaci	49	57	67	70

Tabulka 7.3: Počet komparátorů a zpoždění v testovaných derivacích L-systému s nastavením E_5_4F

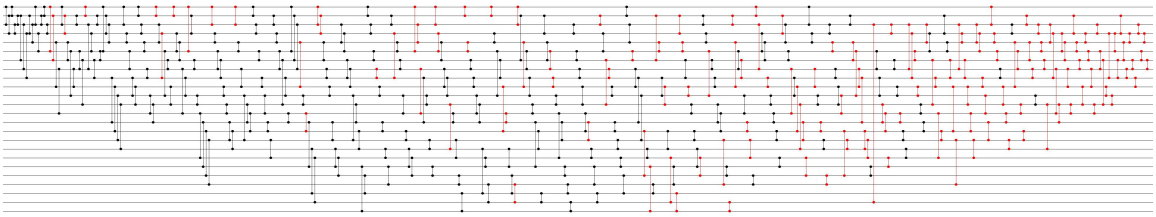
Pro nastavení E_5_4F byl navržen L-systém pro konstrukci řadicí sítě s 24 vodiči (viz obrázek 7.6). Neoptimalizovaná síť s 24 vdiči je na obrázku 7.5. V jednom růstovém kroku byla vždy provedena 1 derivace a přidány 4 vodiče. Následník pravidel obsahoval 5 komponent (C nebo W). Při následném ověření růstu tento systém prokázal schopnost generovat funkční sítě i v dalších 3 derivacích. Maximální velikost sítě generovaná tímto systémem je 36 vstupů. Síť generovaná v další derivaci již nebyla ověřována. V příloze C.2 je tato poslední ověřovaná síť se 36 vstupy vizualizována.

V tabulce 7.3 je zobrazen počet komparátorů a zpoždění řadicích sítí vygenerovaných v jednotlivých derivacích.

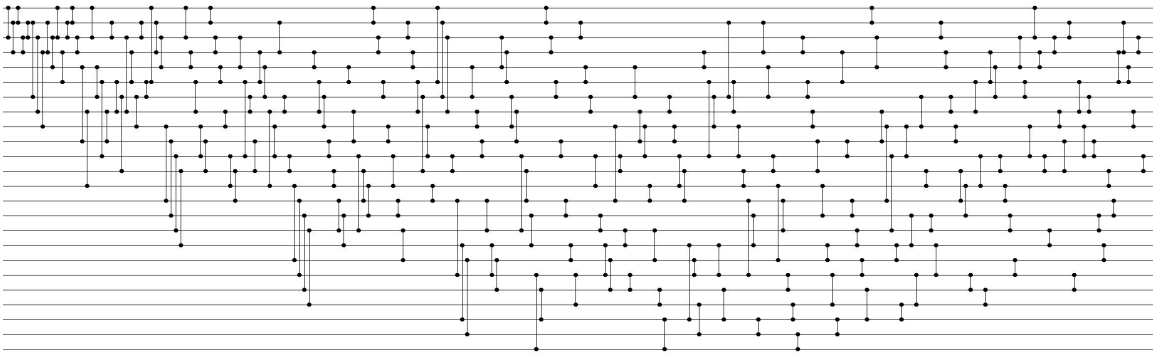
Navržený L-systém s nastavením E_5_4F je tento: $G = (V, \omega, P)$, kde abeceda $V = \{I, O, C, W\}$, axiom $\omega = I(0)C(2)C(1)W(0)O(0)I(1)C(2)W(0)C(1)O(1)I(2)W(0)C(1)W(0)O(2)I(3)W($ a množina pravidel P obsahuje 5 pravidel:

$$
\begin{aligned}
&1) \quad O(i) : (i + i \neq k) \rightarrow C(3\ max\ 2)C(2\ max\ 0)C(0-1)C(1-1)C(0-1)O(i)\\
&2) \quad O(i) : (i * k < 5) \rightarrow C(2-k)W(3\ max\ 4)W(4\%1)W(4*0)C(5\%1)O(i)\\
&3) \quad O(i) : (k\ min\ i\ \neq 1) \rightarrow C(5*1)W(0-0)C(k+1)W(2/1)W(5+1)O(i)\\
&4) \quad O(i) : (i\ max\ k < k) \rightarrow C(2\ max\ k)C(3\ max\ 2)W(2-5)C(5*0)C(3\%0)O(i)\\
&5) \quad O(i) : (k\ max\ i\ nezalezi\ 2) \rightarrow W(0\ max\ 3)C(2/0)C(3*2)W(3*0)W(0\%0)O(i)
\end{aligned}
\tag{7.3}
$$

Posloupnosti komparátorů včetně jejich vizualizace pro následující derivace nejlepších nalezených L-systémů jsou k dispozici na přiloženém CD.



Obrázek 7.5: 5. derivace L-systému s nastavením *E_5_4F*, funkční 24-vstupá řadicí síť před optimalizací



Obrázek 7.6: 5. derivace L-systému s nastavením *E_5_4F*, funkční 24-vstupá řadicí síť po optimalizaci

7.2 Interpretace po vrstvách

Experimentální výsledky prokázaly, že tuto metodu interpretace lze také s úspěchem použít pro evoluční návrh řadicích sítí pomocí L-systémů. Úspěšnost interpretace po vrstvách byla o něco vyšší než u interpretace po vodičích. Průměrná úspěšnost této metody je 16%. Tuto hodnotu opět zkresluje fakt, že úspěšnost této metody s odstraněním vodičů je extrémně nízká.

Výhodou této metody je také to, že generuje více pravidelné vzory komparátorů. To je způsobeno tím, že i při posunu kolidujících komparátorů není komparátor odsunut o tak velký počet vrstev jako u interpretace po vodičích. Vygenerovaná síť tak lépe kopíruje vzory vygenerované L-systémem a nedochází k jejich přílišnému narušení. Nejčastějším vzorem je posloupnost stejně velkých komparátorů pokrývajících postupně celou síť.

7.2.1 Odstranění komparátorů

Pomocí této metody bylo nalezeno pouze jedno řešení. Toto řešení se opět vyznačuje menším počtem komparátorů a menším zpožděním než mají řešení nalezena metodou posunu komparátorů. Ani toto řešení však nebylo schopno v dalších derivacích generovat funkční řadicí sítě.

Opět byly provedeny doplňující experimenty s navýšeným počtem derivací na jeden krok růstu i počet komponent v následníku pravidla. V těchto doplňujících experimentech však nebyl nalezen žádný výsledek.

7.2.2 Posun komparátorů

Tato metoda interpretace se ukázala jako nejúspěšnější. Se stejným nastavením jako v předchozích případech dosahovala úspěšnosti (32%) a bylo zde také objeveno nejvíce rostoucích sítí.

Počet pravidel neměl vliv ani na evoluci ani na ověření dalšího růstu sítí. Evoluce byla opět úspěšnější pro axiom reprezentující funkční řadicí síť (43%) než pro axiom reprezentující prázdnou řadicí síť (21%).

Jak při evoluci tak při následném ověření růstu bylo nejúspěšnější nastavení E. Toto nastavení mělo průměrnou úspěšnost evoluce 66%. V další derivaci bylo 29% nalezených L-systémů schopno generovat funkční síť. Čtvrtina L-systémů nalezených evolucí vygenerovala funkční řadicí síť i ve druhé derivaci. Ve třetí (poslední ověřované) derivaci vygenerovalo funkční řadicí síť 10% L-systémů.

Nejlepší výsledky

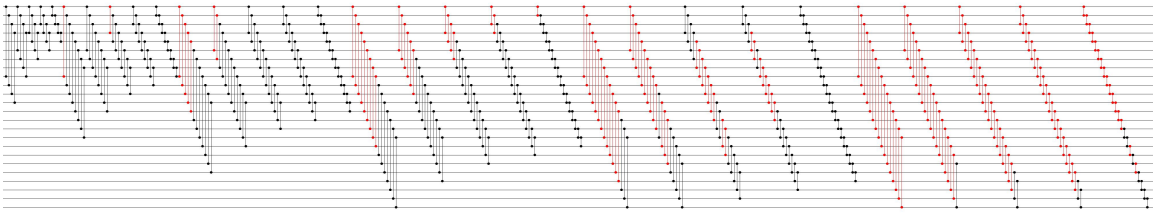
Pro nastavení E_5_4F, E_5_4P, E_12_4F, E_12_4P bylo navrženo několik L-systémů pro konstrukci řadicí sítě s 24 vodiči. V jednom růstovém kroku byla vždy provedena 1 derivace a přidány 4 vodiče. Následník pravidel obsahovala 5 komponent (C nebo W). Při následném ověření růstu 10 z těchto systémů prokázalo schopnost generovat funkční sítě i v dalších 3 derivacích. Maximální velikost sítě generovaná těmito systémy je 36 vstupů. Síť generovaná v další derivaci již nebyla ověřována.

Pro nastavení E_12_4P byl nalezen L-systém s nejzajímavějším vzorem komparátorů. Postupně je vždy pokryta celá aktuální velikost sítě komparátory o délce 8, 5, 3, 2 a 1. Některé z těchto komparátorů jsou kvůli redundanci odstraněny, ale při zobrazení sítě i s těmito komparátory je vzor zcela pravidelný (viz obrázek 7.7). Optimalizovaná řadicí síť o velikosti 24 vodičů vygenerovaná tímto L-systémem znázorněna na obrázku 7.8. V příloze C.3 je vizualizována poslední ověřovaná síť se 36 vstupy. Řadicí sítě generované dalšími rostoucími systémy jsou součástí přiloženého CD. Pro každý z těchto systémů je na CD opět posloupnost komparátorů pro jednotlivé derivace a jejich vizualizace včetně zobrazení redundantních komparátorů.

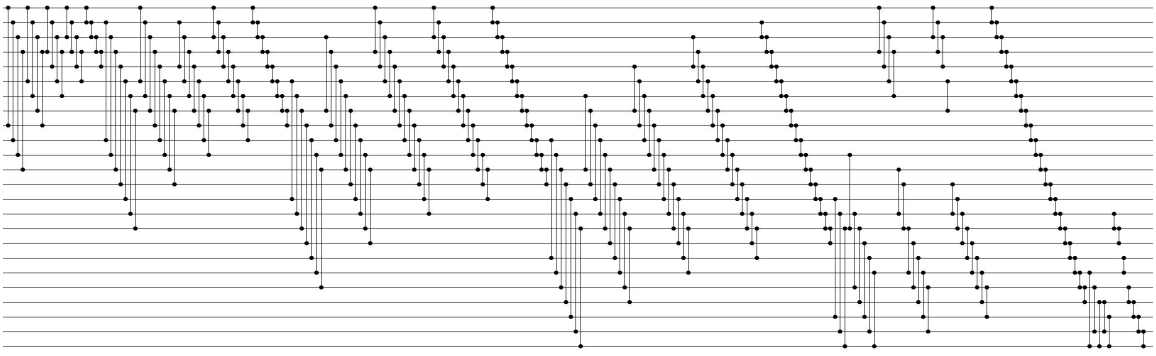
Navržený L-systém s nastavením E_12_4P generující síť na obrázku 7.8 je tento: $G = (V, \omega, P)$, kde abeceda $V = \{I, O, C, W\}$, axiom $\omega = I(0)O(0)I(1)O(1)I(2)O(2)I(3)O(3)$ a

množina pravidel P obsahuje 5 pravidel:

$$
\begin{aligned}
&1) \quad O(i) : (k\ min\ i < 0) \rightarrow W(2\ max\ 0)C(5\ max\ k)W(0/3)C(k*0)C(1\ min\ 1)O(i)\\
&2) \quad O(i) : (i+i < 0) \rightarrow C(k/2)C(2\ max\ 4)C(2*1)W(4-0)C(4\ min\ 1)O(i)\\
&3) \quad O(i) : (i*i\ nezalezi\ 1) \rightarrow C(4*2)C(5\ max\ 2)C(1+3)C(4\ min\ 1)C(4-0)O(i)\\
&4) \quad O(i) : (k-k=k) \rightarrow W(1\%5)C(4/0)W(1+5)W(2\ max\ 1)C(1+1)O(i)\\
&5) \quad O(i) : (i-k>1) \rightarrow C(5\%k)W(5/3)W(0-k)C(2+0)W(0\%0)O(i)\\
&6) \quad O(i) : (k-i\ nezalezi\ 0) \rightarrow W(5+5)W(2*0)W(3*2)C(3*0)C(k*0)O(i)\\
&7) \quad O(i) : (i+i>1) \rightarrow W(5\%0)C(0\ max\ 2)C(4\ min\ 5)C(5\ max\ 0)O(i)C(4\ max\ 0)O(i)\\
&8) \quad O(i) : (k*i<4) \rightarrow W(5-4)C(2\%4)W(1\ min\ 2)W(3-0)C(5/1)O(i)\\
&9) \quad O(i) : (k+i>1) \rightarrow C(0+5)W(3\%3)W(1\ max\ 4)C(2/0)W(2min1)O(i)\\
&10) \quad O(i) : (i\ min\ i>5) \rightarrow C(3\ max\ 4)C(5*1)C(3/k)C(k/1)W(2/0)O(i)\\
&11) \quad O(i) : (k/k=5) \rightarrow W(k+k)W(5-5)W(0-k)W(4*0)W(0-0)\\
&12) \quad O(i) : (k\ max\ i\ nezalezi\ 2) \rightarrow W(4/4)W(5-5)W(4+0)W(0*0))C(3\ min\ 0)O(i)
\end{aligned}
\tag{7.4}
$$



Obrázek 7.7: 5. derivace L-systému s nastavením E_12_4P, funkční 24-vstupá řadicí síť před optimalizací



Obrázek 7.8: 5. derivace L-systému s nastavením E_12_4P, funkční 24-vstupá řadicí síť po optimalizaci

V tabulce 7.4 je zobrazen počet komparátorů a zpoždění řadicích sítí vygenerovaných v jednotlivých derivacích.

Řadicí sítě generované pro nastavení E_5_4F, E_5_4P, E_12_4F, E_12_4P obsahují velmi podobné vzory proto také generují sítě s podobným počtem komparátorů a s

podobným zpožděním. Po srovnání hodnot z tabulky pro konstrukci po vodičích 7.3 s tabulkou pro konstrukci po vrstvách 7.4 je zřejmé, že pro stejné počty vstupů a podobné parametry L-systému je síť generovaná metodou po vstupech více optimální.

velikost sítě	24	28	32	36
počet komparátorů	396	536	696	876
počet komparátorů po optimalizaci	233	318	417	530
zpoždění sítě	161	217	281	356
zpoždění sítě po optimalizaci	107	146	191	243

Tabulka 7.4: Počet komparátorů a zpoždění v testovaných derivacích L-systému s nastavením E_12_4P

Kapitola 8

Závěr

V této práci byly představeny metody, které lze použít k evolučnímu návrhu (částečně) rostoucích řadicích sítí pomocí konkrétního typu přepisovacího systému.

Přepisovací systém použitý pro tuto práci je variantou parametrického L-systému. Evolučním algoritmem je navrhován předem stanovený počet pravidel L-systému včetně jejich podmínek a parametrů.

Pro konstrukci řadicích sítí byli zavedeny 4 komponenty reprezentující vstup, drát, komparátor a výstup. Tyto komponenty mají v abecedě L-systému přiřazeny odpovídající symboly a jsou parametrizované. Parametr vstupní a výstupní komponenty udává vodič na kterém se komponenta nachází. Parametr komparátoru udává jeho velikost.

Interpretaci řetězce vygenerovaného L-systémem lze provádět různými metodami. V této práci byla představena metoda konstrukce řadicí sítě z řetězce L-systému po vodičích. Tato metoda byla inspirována želví grafikou. Želva čte řetězec postupně znak po znaku a v tomto pořadí ihned vkládá požadované komponenty do výsledné sítě. Druhou metodou představenou v této práci byla konstrukce řadicí sítě po vrstvách. Tato metoda byla inspirována konvenčními postupy konstrukce řadicích sítí. Řetězec v tomto případě není interpretován postupně znak po znaku, ale je rozdělen na sekce pro jednotlivé vodiče. V jednom kroku interpretace jsou vždy zpracovávány symboly se stejným indexem v sekci. Pro každou z těchto metod jsou představeny dva způsoby ošetření kolidujících komparátorů. Prvním způsobem je odstranění kolidujících komparátorů ze struktury sítě. Druhým způsobem je posun takovýchto komparátorů do jiné vrstvy.

Výsledky experimentů ukázaly, že evoluce je schopna navrhnout L-systémy, které lze interpretovat jako funkční řadicí sítě. Některé z takto nalezených L-systémů jsou schopny v dalších derivacích generovat funkční sítě s větším počtem vstupů.

Pro evoluci můžeme použít libovolně velké axiomy. Jako axiom lze použít plně funkční, ale například i prázdnou řadicí síť o různém počtu vstupů. V růstovém kroku může řadicí síť růst o libovolný počet vstupů. Pro úspěšnou evoluci/růst je však potřeba provést v každém růstovém kroku dostatečný počet derivací a vložit dostatečný počet komponent. Počet pravidel v L-systému není pro evoluci ani růst klíčový, mezi testovanými 5 a 12 pravidly v jednom jedinci nebyli výrazné rozdíly v úspěšnosti evoluce/růstu.

Nejúspěšnější metodou je konstrukce řadicích sítí po vrstvách s posunem kolidujících komparátorů. Pomocí této metody byli objeveny L-systémy generující 24-vstupé řadicí sítě s pravidelným vzorem komparátorů, které jsou schopny vyrůst až na 36 vstupů. Sítě generované v dalších derivacích nebyly ověřovány, vzhledem k tomu, že časová složitost ověřování funkčnosti řadicích sítí je exponenciální.

Metodou generující nejvíce optimální sítě, které jsou sítě schopny růst je metoda kon-

strukce řadicích sítí po vodičích s posunem kolidujících komparátorů. Zpoždění takových sítí je v poslední ověřované derivaci třikrát menší než u sítí generovaných po vrstvách s posunem kolidujících komparátorů.

Naopak nejnižší úspěšnost měli metody interpretace, které odstřaňovaly kolidující komparátory. Toto omezení sice generovalo o něco optimálnější sítě, avšak úspěšnost evoluce byla menší než 1% a žádná z takto zkonstruovaných sítí nebyla schopna dalšího růstu.

Možným pokračováním této práce by mohla být úprava velmi striktního omezení pro zahození kolidujících komparátorů. Kolidující komparátory by například mohli mít určitý rozah vrstev do kterých se mohou ještě posunout. Pokud by je ani v těchto povolených vrstvách nebylo možno umístit byly by odstraněny. Dalším možným pokračováním této práce je nalézt metody, které by byli schopny v relativně krátké době ověřit funkčnost sítí s vyšším počtem vstupů generované dalšími derivacemi L-systému. Vzhledem k neměnnosti struktury řadicí sítě by bylo vhodné tento výpočet paralelizovat.

Literatura

[1] Batcher, K. E.: *Sorting Networks and Their Applications*. AFIPS '68 (Spring), New York, NY, USA: ACM, 1968, 307–314 s.

[2] Bezák, J.: *Štruktúrny design pomocou celulárnych automatov*. FIT VUT v Brně, 2012, diplomová práce, Brno.

[3] Bidlo, M.: *Evoluční návrh řadicího algoritmu*. FIT VUT v Brně, 2004, diplomová práce, Brno.

[4] Bidlo, M.: *Evolutionary Design of Generic Structures Using Instruction-Based Development*. Faculty of Information Technology BUT, 2010, ISBN 978-80-214-4210-8, 124 s.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=9459

[5] Holland, J.: *Adaptation in natural and artificial systems*. Cambridge, Mass.: MIT Press, první vydání, 1992, ISBN 02-625-8111-6.

[6] Hornby, G. S.; Pollack, J. B.: The advantages of generative grammatical encodings for physical design. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress*, 2001: s. 600 – 607, iSBN 0-7803-6657-3.
URL http://www.demo.cs.brandeis.edu/papers/hornby_cec01.pdf

[7] Hornby, G. S.; Pollack, J. B.: Evolving L-Systems To Generate Virtual Creatures. *Computers and Graphics*, ročník 25, č. 6, 2001: s. 1041 – 1048.
URL http://www.demo.cs.brandeis.edu/papers/hornby_cag01.pdf

[8] Hynek, J.: *Genetické algoritmy a genetické programování*. Grada Publishing, a.s., 2008, iSBN 978-80-247-2695-3.

[9] Knuth, D. E.: *The art of computer programming*. Addison Wesley, 1973, volume 3: Sorting and Searching.

[10] Kumar, S.; Bentley, P. J.: *On Growth, Form and Computers*. Elsevier Academic Press, 2003, iSBN 0-12-428765-4.

[11] Kvasnička, V.; pospíchal, J.; Tiňo, P.: *Evolučné algoritmy*. Vydavateľstvo STU v Bratislave, 2000, iSBN 80-227-1377-5.

[12] Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, ročník vol. 18, č. issue 3, 1968: s. 280–299, ISSN 00225193, doi:10.1016/0022-5193(68)90079-9.
URL http://linkinghub.elsevier.com/retrieve/pii/0022519368900799

[13] Lindenmayer, A.: Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, ročník vol. 18, č. issue 3, 1968: s. 300–315, ISSN 00225193, doi:10.1016/0022-5193(68)90080-5.
URL http://linkinghub.elsevier.com/retrieve/pii/0022519368900805

[14] O'Neill, M.; Brabazon, A.: Evolving a Logo Design using Lindenmayer Systems, Postscript & Grammatical Evolution. *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)*, 2008: s. 3788 – 3794, iSBN 978-1-4244-1822-0.
URL http://ncra.ucd.ie/logodesign.pdf

[15] Prusinkiewicz, P.; Lindenmayer, A.: *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990, iSBN 978-0-387-94676-4.
URL http://algorithmicbotany.org/papers/abop/abop.pdf

[16] Sekanina, L.; Bidlo, M.: Evolutionary Design of Arbitrarily Large Sorting Networks Using Development. *Genetic Programming and Evolvable Machines*, ročník 6, č. 3, 2005: s. 319–347, ISSN 1389-2576.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=7742

[17] Sekanina L. a kolektiv: *Evoluční hardware*. Academia, 2009, iSBN 978-80-200-1729-1.

Přílohy

Seznam příloh

Příloha A

Obsah CD

Na přiloženém CD se ve složce **EA_src** nacházejí zdrojové kódy evolučního algoritmu a skript pro vizualizaci výsledku evolučního návrhu. Složka **nejlepsi_vysledky** obsahuje podsložky pro nejlepší výsledky provedených experimentů. V těchto podsložkách jsou soubory s posloupnostmi komparátorů a vizualizace řadicích sítí pro jednotlivé testované derivace nejlepších nalezených L-systémů. Složka **DP_tex** obsahuje zdrojové kódy pro vysázení diplomové práce v programu LaTeX a výsledný soubor ve formátu pdf.

A.1 Zdrojové kódy pro EA

Zdrojové kódy jsou rozděleny do tří složek. První složka obsahuje zdrojové kódy pro evoluci L-systémů. Druhá složka obsahuje zdrojové kódy pro ověření růstu řadicí sítě v dalších derivacích. Poslední složka obsahuje program schopný pravidla evolučně navrženého L-systému převést do čitelné podoby. Ve složce **EA_src** se také nachází skript napsaný v jazyce python `generate_networks.py`, který vizualizuje navržené řadicí sítě.

Evoluční algoritmus i ověřování růstu je rozděleno dle jednotlivých variant interpretace do čtyř následujících podsložek:

- **vodice_posun** - složka obsahující EA/ověření růstu s interpretem pro konstrukci sítí po vodičích s posunem kolidujících komparátorů
- **vodice_zahozeni** - složka obsahující EA/ověření růstu s interpretem pro konstrukci sítí po vodičích se zahozením kolidujících komparátorů
- **vrstvy_posun** - složka obsahující EA/ověření růstu s interpretem pro konstrukci sítí po vrstvách s posunem kolidujících komparátorů
- **vrstvy_zahozeni** - složka obsahující EA/ověření růstu s interpretem pro konstrukci sítí po vrstvách se zahozením kolidujících komparátorů

Každá z těchto složek obsahuje:

- složku s předdefinovanými axiomy `axioms`
- soubor `Makefile` pro kompilaci
- konfigurační soubor `params.h` a hlavičkové soubory `main.h`, `eaparts.h`, `interpreter.h`, `zeroone.h`
- soubory se zdrojovými kódy `main.cpp`, `eaparts.cpp`, `interpreter.cpp`, `zeroone.cpp`

Příloha B

Návod k použití programu

Programy pro evoluci, simulaci růstu i vizualizaci sítí jsou spustitelné na platformě Linux. Program používá pouze běžně dostupné knihovny. Funkčnost programu byla testována na superpočítačích Anselm a Salomon, na PC s operačním systémem Ubuntu 12.04 a na školním serveru Merlin.

B.1 Spuštění programu pro evoluci řadicí sítě

1. Vyberte složku s požadovaným interpretem.
2. V kofiguračním souboru `params.h` zadejte požadované parametry pro navrhovaný L-systém a EA.
3. Přeložte program pomocí příkazu `make`.
4. Po spuštění programu v příkazové řádce proběhne výpočet. Výsledek je vytisknut do terminálu. Pro další ověření růstu je vhodné výstup přesměrovat do souboru.

B.2 Spuštění programu pro ověření růstu řadicí sítě

1. Vyberte složku s požadovaným interpretem.
2. V konfiguračním souboru `params.h` zadejte parametry pro L-systém a EA, které byli použity při evoluci a parametry určující soubor s ověřovaným jedincem a požadovanou velikost.
3. Přeložte program pomocí příkazu `make`.
4. Po spuštění programu v příkazové řádce proběhne výpočet. Výsledek je vytisknut do terminálu.

B.3 Vizualizace

Pomocný program pro vizualizaci se spustí `python generate_networks.py`. Skript vykreslí všechny sítě ze složky inputs a výsledné obrázky umístí do složky outputs. Pro spuštění tohoto skriptu je potřeba mít nainstalovány knihovny Tkinter, Image a ConfigParser.

B.4 Popis parametrů v konfiguračním souboru

Následující parametry slouží ke konfiguraci EA:

- `PMUT` - pravděpodobnost mutace v %
- `PCROSS` - pravděpodobnost křížení v %
- `POPSIZE` - velikost populace
- `MAXGENERATIONS` - maximální počet generací
- `RULESCOUNT` - počet pravidel v jednom jedinci
- `RIGHTLENGTH` - počet komponent C a W v následníku pravidla

Následující parametry slouží ke konfiguraci navrhovaného L-systému:

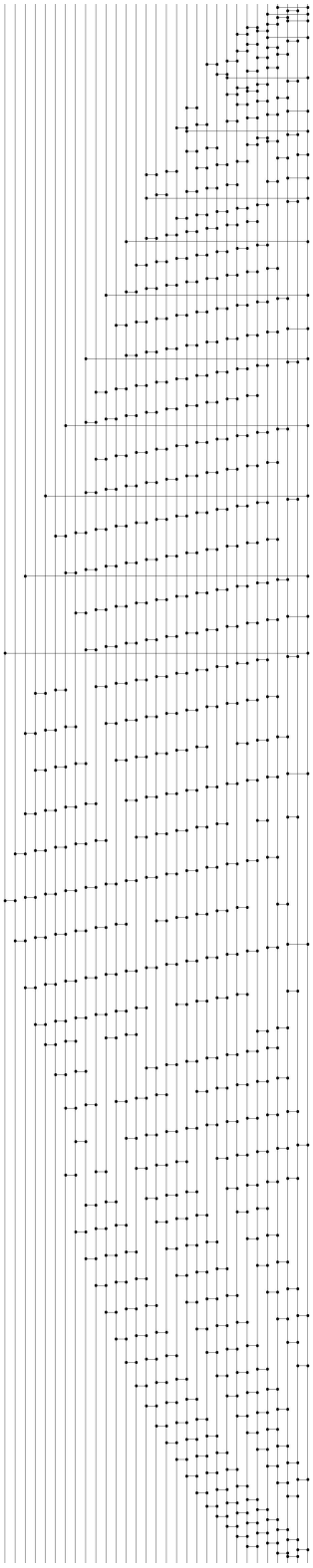
- `NETSIZE` - velikost generované sítě
- `FUNCSIZE` - velikost již funkční sítě (parametr pro růst, zabraňuje opětovnému ověřování menších sítí)
- `NETSTEP` - velikost kroku vývoje (kolik vodičů se přidá v jednom kroku)
- `ITERSTEP` - počet iterací na jeden krok vývoje
- `AXIOMSIZE` - počet vodičů v axiomu
- `AXIOMFILE` - soubor obsahující axiom
- `INDIVIDUALFILE` - soubor obsahující jedince (parametr pro růst)

Pro parametr `NETSIZE` musí platit `NETSIZE = AXIOMSIZE + x*NETSTEP`, kde $x \in \mathbb{N}$. Pokud je `AXIOMFILE` nastaven na prázdný řetězec vygeneruje se axiom s prázdnou sítí o velikosti `AXIOMSIZE`.

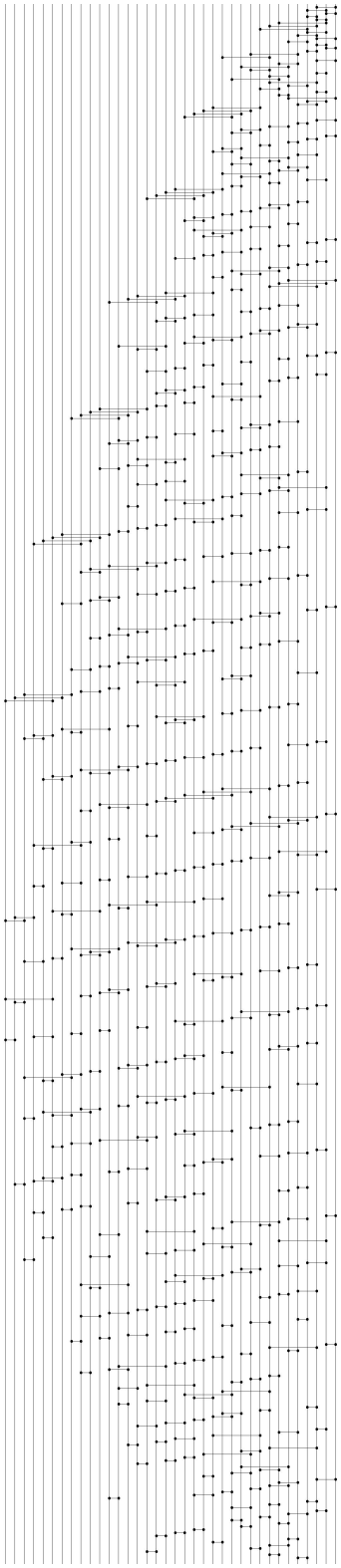
Příloha C

Vizualizace poslední derivace nejlepších výsledků

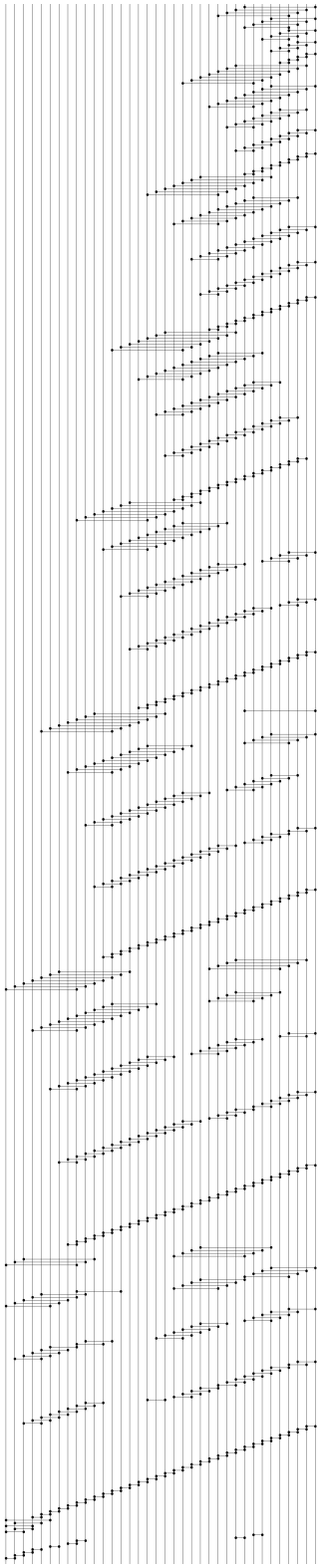
Na následujících stránkách jsou zobrazeny poslední vizualizované derivace systémů představených v kapitole 7. Nultý vstup se nachází vpravo, poslední vsup vlevo. Zobrazené sítě jsou již po optimalizaci odstraněním redundantních komparátorů. Neoptimalizované sítě stejně jako sítě z dalších derivací jsou k dispozici na přiloženém CD.



Obrázek C.1: 18. derivace L-systému s nastavením B_5_4P, funkční 31-vstupá řadicí síť po optimalizaci



Obrázek C.2: 8. derivace L-systému s nastavením E_5_4F, funkční 36-vstupá řadicí síť po optimalizaci



Obrázek C.3: 8. derivace L-systému s nastavením E_12_4P, funkční 36-vstupá řadicí síť po optimalizaci