VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYUŽITÍ PŘIBLIŽNÉHO POČÍTÁNÍ V OBLASTI ZPRACOVÁNÍ OBRAZU

APPLICATION OF APPROXIMATE COMPUTING IN IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **PETR HRUDA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Hruda Petr, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Využití přibližného počítání v oblasti zpracování obrazu**
Application of Approximate Computing in Image Processing
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s problematikou přibližného počítání, možnostmi návrhu aproximativních obvodů a jejich využití v oblasti zpracování obrazu.
2. Zvolte vhodný algoritmus (algoritmy) z oblasti zpracování obrazu a v tomto identifikujte části vhodné k aproximaci.
3. Navrhněte techniku pro substituci těchto částí algoritmu přibližnými implementacemi výpočetních operací.
4. Proveďte sady experimentů s využitím různých konfigurací systému z bodů 2 a 3 a sledujte vybrané parametry výsledného řešení.
5. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Podle pokynů vedoucího projektu.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 a 2 zadání, demonstrace prototypu systému z bodu 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bidlo Michal, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 20. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato semestrální práce se zabývá aplikací techniky přibližného počítání na oblast zpracování obrazu. Konkrétně je aproximace uplatněna na adaptivní prahování obrazu. Byly využity dva přístupy, návrh nového systému za poskytnutí aproximovaných součástek a aproximace existujícího algoritmu. Byl zkoumán výsledný vliv na kvalitu prahování. Experimentální vyhodnocení prvního přístupu vykazuje zlepšení kvality prahování s rozumným stupněm aproximace poskytnutých součástek. Snížena je i plocha, kterou navržené řešení zabírá. Vyhodnocení druhého přístupu vykazuje zhoršení kvality s využíváním aproximací a tento přístup je tedy označen za nevhodný.

Abstract

This master's thesis focuses on approximate computing applied to image processing. Specifically, the approximation is applied to adaptive thresholding. Two approaches were used, the design of a new system using approximated components and the approximation of an existing algorithm. The resulting effect on thresholding quality was investigated. Experimental evaluation of the first approach shows quality improvements of thresholding with usage of aproximated components. Also, area of found aproximated solutions is smaller. Evaluation of the second approach shows worse quality of thresholding with usage of aproximated components. The second approach is then declared inappropriate.

Klíčová slova

Přibližné počítání, adaptivní prahování, CGP, NSGA-II, aproximace algoritmu, multikriteriální optimalizace, genetické programování

Keywords

Approximate computing, adaptive thresholding, CGP, NSGA-II, algorithm approximation, multiobjective optimalization, genetic programming

Citace

HRUDA, Petr. *Využití přibližného počítání v oblasti zpracování obrazu.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Využití přibližného počítání v oblasti zpracování obrazu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Michala Bidla. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Hruda
3. června 2020

Poděkování

Tímto bych rád poděkoval vedoucímu panu doktorovi Bidlovi, který ochotně poskytoval odbornou pomoc a vedení i přes veškeré překážky letošního roku. Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z podpory Velkých infrastruktur pro výzkum, experimentální vývoj a inovace v rámci projektu "IT4Innovations národní superpočítačové centrum - LM2015070"

Obsah

Kapitola 1

Úvod

V současné době je většina výpočtů prováděna na zařízeních, která jsou citlivá na celkovou spotřebu elektrické energie. Například mobilní telefony, zařízení kontrolující provoz na silnicích, či velká datová centra. Je žádoucí snižovat spotřebu energie a tím prodloužit životnost mobilní baterie, či zajistit běh kontrolních silničních zařízení pouze s pomocí solárních panelů. Pro velká datová centra je spotřeba energie důležitým faktorem celkové ceny provozu. Spotřeba je jedním z důvodů, proč jsou v praxi užívány optimalizace algoritmů, hardwarové akcelerace, nebo např. omezení „*run-time*“ programu. Vektorové procesory, FPGA čipy, grafické procesory, atd. jsou využívány ve společné snaze optimalizovat výkon a snížit tak spotřebu energie [7].

Jednou z technik, která se zabývá problematikou snižování spotřeby elektrické energie je tzv. přibližné počítání. Principem tohoto přístupu je navrhnout systém, který produkuje přibližné výsledky s přípustnou chybou. Zavedení určité chyby do výpočtu dovoluje využít součástky s nižší přesností. Ty mají zpravidla nižší spotřebu, než přesné řešení. Jedním z příkladů může být ztrátová komprese. Výsledkem této komprese je přibližný obrázek. Návrh takovéto komprese je založen na odebrání informací, které lidské oko ztěží (pokud vůbec) rozpozná. Z tohoto příkladu se lze domnívat, že technika zavádění přibližných výpočtů je již tradičním přístupem v návrhu algoritmů. Tyto techniky však byly navrženy manuálně na základě zkušeností s daným problémem. Cílem aktuálního výzkumu přibližného počítání je tento proces automatizovat. Jinými slovy, záměrem je automaticky navrhnout efektivnější systém pro daný problém, který produkuje přibližené výsledky v rámci přípustné chyby [7].

Úspora spotřeby systému není jediným cílem přibližného počítání. Na současném čipu produkovaném moderními technologiemi je mnoho komponent, které jsou již od výroby nepřesné. Výroba naprosto přesných součástek je drahá. Je tedy třeba, brát tyto nepřesnosti v úvahu již při návrhu algoritmu.

Cílem této práce je aplikovat techniku přibližného počítání v odvětví zpracování obrazu. Konkrétně byla vybrána oblast adaptivního prahování psaného textu. V této práci byl testován vliv aproximace existujícího algoritmu a samotný návrh nového algoritmu s poskytnutím již aproximovaných součástek. Obě tyto úlohy byly řešeny pomocí evolučních algoritmů.

Kapitola 2

Evoluční algoritmy

Evoluční a genetické optimalizační algoritmy využívají modely evoluce v přírodě. Základní úloha spočívá v řešení optimalizačního problému. Cílem optimalizačního problému může být hledání minima dané funkce. Jedná se pak o problém minimalizace. Pokud jde o hledání maxima dané funkce, jedná se o problém maximalizace. Tyto dva problémy lze mezi sebou jednoduše převádět. Jelikož oba postupy fungují principiélně stejně, budou níže popsané principy ilustrovány na problému minimalizace. Optimum x_{opt} lze mít definováno takto:

$$x_{opt} = arg\ min\ f(x), \tag{2.1}$$

pro nějakou funkci $f : R^k \rightarrow R^+$, která zobrazuje vektory reálných čísel délky k na kladné reálné číslo. Funkce f je v tomto případě nazývána jako *účelová funkce* [5]. Evoluční algoritmy jsou využívány především u problémů, kdy je stavový prostor rozsáhlý a s velkým množstvím lokálních minim. V těchto případech přesné metody nedokáží nalézt řešení v rozumném čase. Přednost proto dostávají metody, které sice nalezení optima negarantují, ale výsledkem je dostatečně kvalitní řešení nalezené v rozumném čase.

Evoluční algoritmy prohledávají stavový prostor tak, že náhodně mění navrhované řešení a neužívají derivaci optimalizované funkce, pouze funkční hodnoty samotné. V případě nalezení lepšího řešení nahrazuje toto v evoluci řešení horší. Síla všeobecnosti evolučních algoritmů spočívá v tom, že je lze použít na optimalizaci prakticky jakékoliv funkce, jejíž hodnota je dostatečně rychle zjistitelná [8].

Tyto algoritmy jsou inspirovány Darwinovou evoluční teorií, konkrétně těmito složkami [8]:

- *Přirozený výběr* – Kvalitnější jedinci s vysokou pravděpodobností přežijí slabé. Díky tomu se celá populace lépe adaptuje na okolní prostředí.
- *Náhodný genetický drift* – Náhodné události jedinců ovlivňují celou populaci. Např. náhodná mutace, náhodná smrt kvalitního jedince. Tyto události jsou významné především pro populace s relativně nízkým počtem jedinců.
- *Dědičnost* – Vlastnosti rodičů jsou předány potomkům. A nejsou tak ztraceny po úmrtí.
- *Speciace* – Výsledkem adaptace na prostředí vznikají nové druhy.

Základním pojmem těchto algoritmů je *populace chromozomů*. V *chromozomu* α je zakódován bod ve stavovém prostoru daného problému, typicky pomocí řetězce symbolů.

$$\alpha = (\alpha^1, \alpha^2, \ldots, \alpha^k) \tag{2.2}$$

Symboly α^i mohou být například dle daného problému z množiny reálných čísel. Hodnota k je obvykle pro daný problém fixní.

Populace P obsahuje chromozomy – řetězce α.

$$P = \{\alpha_1, \alpha_2, ..., \alpha_n\} \tag{2.3}$$

Funkce f, která každému chromozomu přiřadí hodnotu *fitness*, je nazývána *fitness funkce*.

Hodnota *fitness* určuje, jak je dané kandidátní řešení kvalitní. Chromozomu, který odpovídá kvalitnímu řešení, je přiřazena nízká fitness, chromozomům zastupujícím horší řešení zase fitness vyšší.

Dále jsou chromozomům populace P definovány dvě základní operace: *křížení* a *mutace*.

Mutací chromozomu se rozumí náhodný výběr symbolu z řetězce (*genu*) a jeho náhrada jiným náhodně vygenerovaným symbolem. Díky mutaci mohou do evoluce proniknout i chromozomy obsahující symboly, které se zde dříve vůbec nevyskytovaly.

Operací *křížení* O_{CROSS} je proces, kdy je ze dvou rodičovských chromozomů vygenerován nový chromozom potomka.

$$O_{CROSS}(\alpha, \beta) \rightarrow \gamma, \tag{2.4}$$

kdy α, β představují chromozomy rodičů a γ chromozom potomka.

Výběr jedinců z populace P, kteří za účelem reprodukce podstoupí křížení, je řízen mechanismem *selekce*. Lze vybírat na základě fitness, často bývá zahrnut i stochastický prvek.

Generací se rozumí jedna iterace evolučního algoritmu. V rámci jedné generace dojde k selekci rodičů, jejich křížení, aplikování mutace a následně výběr nové populace (mechanismus nahrazování jedinců).

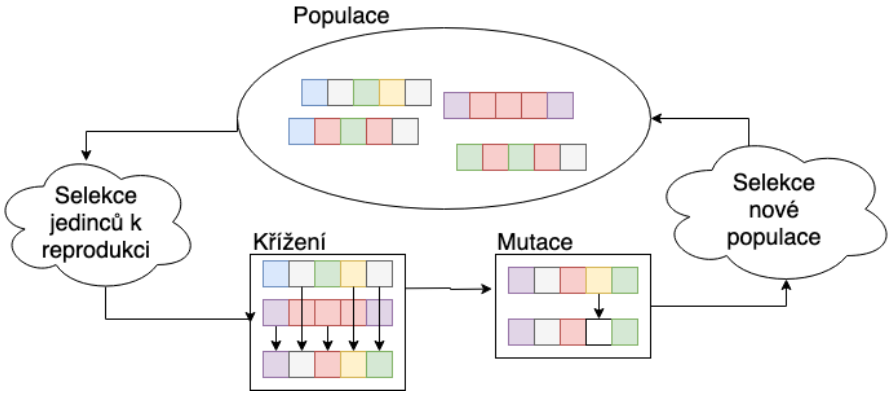
2.1 Evoluční strategie

Evoluční strategie (dále jen ES) byla navržena v 60. letech jako experimentální optimalizační metoda [1]. V té době byla využívána především na úlohy s povahou reálných čísel, lze ji ale aplikovat i na aplikace s diskrétní reprezentací. Evoluční strategie definuje symboly μ a λ. Symbol μ v notaci vyjadřuje velikost populace, λ určuje počet potomků vzniklých v jedné generaci. Na ES bude detailněji vysvětlen princip evolučních algoritmů. Základní myšlenka lze pozorovat na algorimu 1 [1].

Algorithm 1: Princip evoluční strategie

```
Náhodně vytvoř prvotní populaci {x_1, x_2, ... x_μ} rodičovských vektorů takových, že
  každý vektor x_i má tvar x_i = x_i^1, ... x_i^n;
Každému jedinci z populace přiřaď fitness na základě kvality řešení, které
  reprezentuje;
while Ukončující podmínka do
    while Počet vygenerovaných potomků < λ do
        Náhodně vyber rodiče z populace {x_1, ..., x_μ};
        Vytvoř potomka aplikováním křížení na rodiče;
        Aplikuj mutaci na potomka;
    end
    Každému potomku přiřaď fitness na základě kvality řešení, které reprezentuje;
    Vyber μ nejlepších jedinců, kteří budou tvořit následující generaci;
end
```

Jednotlivé části ES jsou znázorněny na obrázku 2.1.



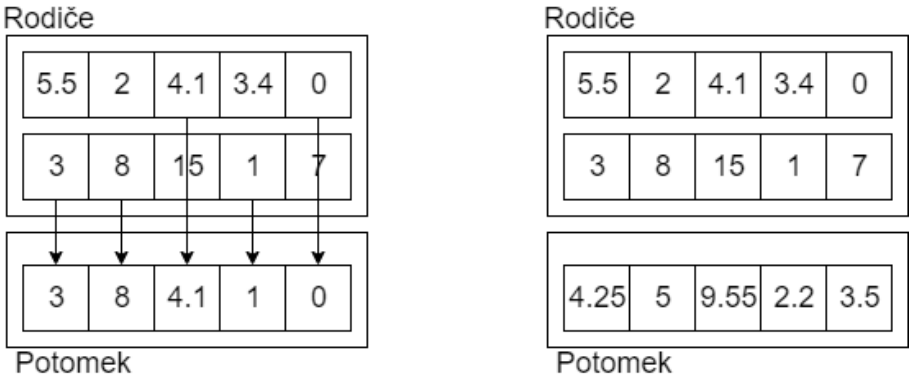
Obrázek 2.1: Princip implementace evoluční strategie. Z populace jsou vybráni jedinci k reprodukci a je na ně aplikováno křížení. Dále je náhodně provedena mutace. Z nových (případně i starých) jedinců je vybrána nová populace.

2.1.1 Křížení

Proces křížení rodičovských chromozomů může být v ES přistupováno několika způsoby. Pro objasnění principů nyní uvažujme případ, kdy jsou chromozomy kódovány jako vektory R^n. Jednotlivé přístupy lze rozdělit na *lokální* a *globální* křížení [8].

Uvažujeme-li křížení lokální, je vytvářen celý potomek ze stejných rodičovských chromozomů (nyní uvažujme dva, obecně jich může být i více). Proces je demonstrován na obrázku 2.2. Výsledného potomka lze vytvořit jako náhodnou kombinaci jednotlivých hodnot rodičovských chromozomů (*diskrétní* křížení), nebo jako agregaci rodičovských hodnot (*intermediární* křížení). Pro definovaný problém v R^n lze využít např. aritmetický průměr odpovídajících hodnot [2].

Rozdíl u globálního křížení spočívá oproti lokálnímu v tom, že pro každou hodnotu chromozomu potomka jsou vybrány náhodné rodičovské chromozomy. Samotná hodnota je pak počítána obdobně jako u křížení lokálního (diskrétní, či intermediární způsob).



Obrázek 2.2: Princip lokálního křížení. Vlevo diskrétní, vpravo intermediární (aritmetický průměr).

2.1.2 Mutace

Operátor mutace bývá specificky navržen dle daného problému. Příkladem nechť je opět problém, pro který jsou chromozomy reprezentovány jako vektory R^n. Předpokládejme, že společně se samotným chromozomem α

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n), \tag{2.5}$$

máme k dispozici vektor odchylek σ

$$\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n), \tag{2.6}$$

udávající odchylky pro konkrétní položky chromozomu α. Pak lze mutaci aplikovat tak, že položku chromozomu α_i změníme o náhodně vygenerovanou hodnotu z normálního rozložení upravenou koeficientem σ_i.

$$\alpha_i' = \alpha_i + \sigma_i * N(0,1); i \in \langle 1, n \rangle, \tag{2.7}$$

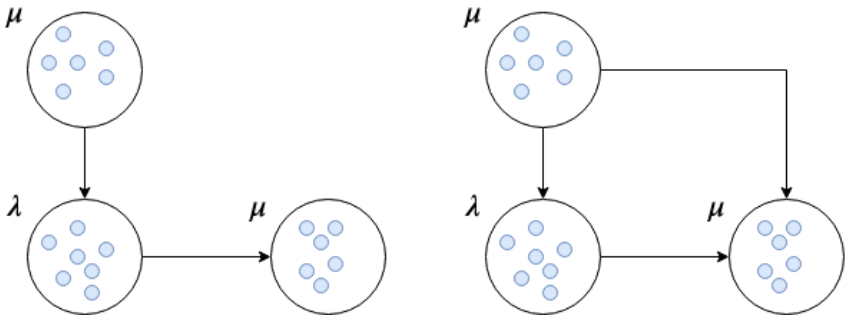
kde α_i' je nová hodnota genu po mutaci a $N(0,1)$ je náhodný generátor s normálním rozložením se středem v nule [2].

2.1.3 Selekce nové populace

ES zavádí speciální notaci, která definuje, jakým způsobem probíhá tvoření nové generace. Budeme uvažovat tyto dvě varianty [1]:

- $(\mu + \lambda)$,
- (μ, λ).

Rozdíl těchto dvou variant lze pozorovat na obrázku 2.3. Tzv. plus-selekce utváří novou populaci výběrem z původních rodičů a jejich potomků, zatímco čárka-selekce pouze z potomků.



Obrázek 2.3: Různé varianty vytváření nové populace ES. Vlevo (μ, λ), kdy je nová populace vybrána pouze z potomků. Vpravo $(\mu+\lambda)$, kdy je nová populace vybrána z potomků i rodičů.

Samotný výběr jedinců kteří postoupí do další generace pak probíhá na základě hodnoty fitness. Například je zvoleno μ jedinců s nejnižší fitness.

2.1.4 Selekce jedinců k reprodukci

Notace ES zavádí parametr ρ, který specifikuje, kolik rodičů populace postoupí do procesu reprodukce. Notace zavedená v sekci 2.1.3 je pak rozšířena na $(\mu/\rho+\lambda)$ respektive $(\mu/\rho,\lambda)$. Samotný výběr pak probíhá buď zcela náhodně, nebo například kvazináhodně pomocí „*ruletty*“ [5].

2.1.5 Samoadaptace

Evoluční strategie mohou do svého běhu zahrnovat tzv. *samoadaptaci* parametrů. Jde o mechanismus, který v běhu evoluce mění také samotné parametry evoluce. Uvažujme podobný problém jako výše u mutace (sekce 2.1.2). Je-li do ES zahrnuta samoadaptace, bývá vektor odchylek σ zahrnut do samotného chromozomu. Chromozom α pak má tvar [5]:

$$\alpha = \langle(\alpha_1, \alpha_2, \ldots, \alpha_n), (\sigma_1, \sigma_2, \ldots, \sigma_n)\rangle$$

Vývoj odchylek pak probíhá stochasticky před jejich uplatněním v mutaci hodnot α_i. Podrobnosti tohoto mechanismu nejsou dále rozebírány, jelikož tento mechanismus v práci užit nebyl.

2.2 Genetické programování

Hlavní rozdíl genetického programování (dále jen GP) od ES spočívá v tom, že chromozomy nekódují pouze parametry daného problému, ale složitější struktury – funkce. V kontextu genetického programování se pod pojmem funkce rozumí výraz obsahující proměnné, konstanty, základní aritmetické operace a elementární funkce. Tuto funkci lze také chápat jako spustitelný program [8].

Díky této reprezentaci chromozomů jsou užity také jiné techniky pro provádění operátorů mutace a křížení.

2.2.1 Reprezentace funkce

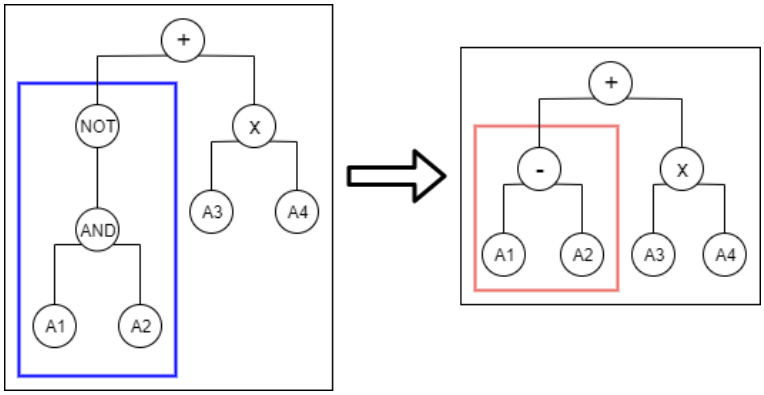
Funkce jsou reprezentovány pomocí syntaktických stromů. Příklad takového stromu lze pozorovat na obrázku 2.4 (černě ohraničená část vlevo). Horní vrchol stromu se nazývá kořen stromu. Interpretace stromu se provádí zdola nahoru [8].

2.2.2 Fitness chromozomu

Jelikož chromozomy v GP reprezentují funkce, je ohodnocení chromozomu prováděno následovně. Je třeba mít k dispozici trénovací množinu referenčních výsledků pro definované vstupy. Nad těmito vstupy se provede funkce, kterou chromozom zastupuje. Hodnota fitness je následně odvozena od rozdílu referenčních výstupů a výstupů chromozomu [8].

2.2.3 Mutace

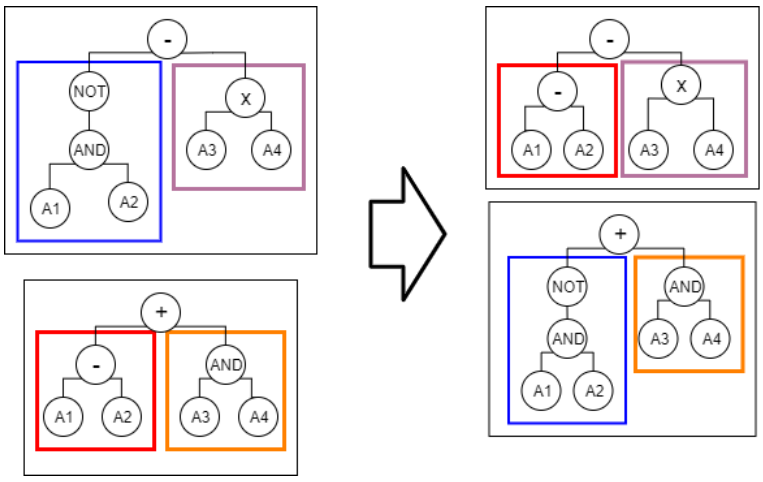
Princip mutace lze pozorovat na obrázku 2.4. Myšlenka spočívá v nahrazení náhodně zvoleného podstromu novým – náhodně vygenerovaným podstromem [8]. Například, náhodně zvolíme uzel NOT. Tento uzel a celý jeho podstrom (obrázek 2.4 – modře ohraničená část) nahradíme náhodně vygenerovaným podstromem, jak je vidět na červeně ohraničené části v obrázku 2.4.



Obrázek 2.4: Ukázka mutace chromozomu. Modrý podstrom je nahrazen náhodně vygenerovaným červeným podstromem.

2.2.4 Křížení

Základní myšlenka křížení spočívá v náhodném výběru podstromů obou rodičovských chromozomů a jejich prohození (obrázek 2.5) [8].



Obrázek 2.5: Křížení chromozomů. U obou rodičovských chromozomů (černě ohraničené části vlevo) jsou náhodně vybrány podstromy. Tyto podstromy jsou pak mezi rodičema prohozeny. Zde byl prohozen modrý podstrom s podstromem červeným. Vznikli tak noví jedinci znázornění vpravo.

2.2.5 Symbolická regrese

Základní aplikací GP je *symbolická regrese* [8]. Vstupem symbolické regrese je dvojice (X, Y) vstupů $X = X_1, X_2, \ldots, X_n$ a výstupů $Y = Y_1, Y_2, \ldots, Y_n$. Cílem symbolické regrese je nalézt pomocí GP předpis funkce F takové, že $Y_i' = F(X_i), i \in \langle 1, n \rangle$, přičemž je minimalizován rozdíl $|Y_i - Y_i'|$ [8].

2.3 Kartézské genetické programování

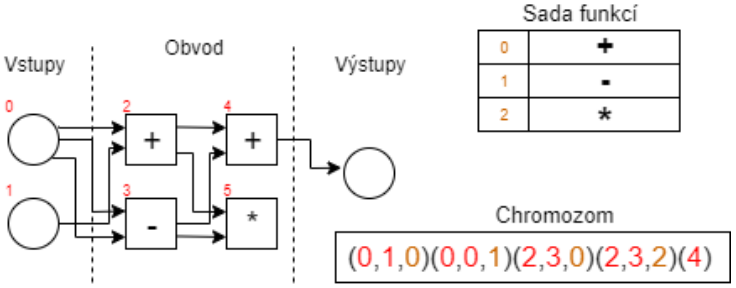
Narozdíl od výše uvedených, kartézské genetické programování (dále referováno jako CGP) není evolučním algoritmem. Jedná se pouze o reprezentační techniku, která je v evoluci využita. Konkrétně se CGP využívá k reprezentaci programů, pomocí sátě propojených funkčních bloků. Podrobnější popis se nachází v následující sekci.

2.3.1 CGP reprezentace

Ve standardním pojetí CGP jsou programy reprezentovány pomocí acyklických orientovaných grafů [8]. Tyto grafy jsou reprezentovány pomocí mřížky výpočetních uzlů. Mřížka je dvou-dimenzionální – odtud pojem *kartézské*. Mřížka s počtem řádků R a počtem sloupců C bude v dále v této práci vyjadřována jako mřížka RxC.

Každý uzel v mřížce má svou adresu dle svého umístění. Princip adresování je možno sledovat na obrázku 2.6. Má-li hledaný program n vstupů, začíná adresace výpočetních uzlů celočíselnou hodnotou n. Hodnoty $\langle 0, n-1 \rangle$ jsou rezervovány právě pro programové vstupy.

Každý výpočetní uzel představuje určitou funkci a je zakódován několika celočíselnými hodnotami (geny). Každý uzel obsahuje *funkční gen*, který odkazuje na jednu z uživatelem definovaných výpočetních funkcí. Ostatní geny popisují propojení uzlu s ostatními. Jelikož jde o orientovaný graf, je rozlišováno mezi geny popisujícími vstup a geny popisujícími výstup. Vstup uzlu je reprezentován celočíselnou hodnotou, která udává adresu uzlu ze kterého vede vstup, nebo adresu programového vstupu. Počet vstupů uzlu je závislý na funkci, kterou uzel reprezentuje. Výstupní gen je posledním genem chromozomu. Udává adresu uzlu, jehož výstup je zároveň výstupem programu, který CGP reprezentuje. Pokud se uzel nachází ve sloupci c, je výstup možno propojit s uzlem, který se nachází vpravo od něj. Tedy alespoň ve sloupci $c+1$.Maximální vzdálenost sloupců mezi kterými může existovat propoj je dán parametrem *l-back*. Odtud vyplývá, že CGP neumožňuje vytvářet zpětné vazby mezi funkcemi.



Obrázek 2.6: Ukázka chromozomu CGP a znázornění jeho příslušného obvodu. Červené hodnoty chromozomu reprezentují adresy připojených bloků. Hnědé hodnoty udávají, která funkce je v bloku použita. Kolečka v obvodu představují globální vstupy CGP a výstup CGP. Čtverečky pak reprezentují jednotlivé bloky.

2.3.2 Evoluční strategie s CGP

CGP není evolučním algoritmem, ale pouze způsobem reprezentace kandidátního řešení. Ukázalo se, že nejlépe CGP funguje ve spojení s evoluční strategií [15]. Chromozom je zakódován dle výše uvedeného popisu. V každé generaci je vybrán pouze jeden chromozom (např. s nejnižší fitness), na který je následně uplatněna mutace. Jde tedy o ES $(1+\lambda)$.

Mutace se provádí změnou náhodného genu na jinou náhodnou hodnotu. Tato hodnota však musí být platná z hlediska definovaného CGP. Zpravidla je toto náhodné generování prováděno následovně. Při inicializaci programu je pro každou pozici genu v chromozomu definováno, jaké hodnoty může obsahovat. Samotné generování pak probíhá nad touto definovanou množinou.

2.4 Multikriteriální optimalizace

Výše vysvětlované principy jednotlivých evolučních přístupů byly uvažovány s fitness funkcí, která bere v potaz pouze jedno kritérium kvality. Existuje však mnoho problémů, kdy je kritérií při vyhodnocování kvality potřeba více. Často pak jsou tato kritéria protichůdná a vylepšení jednoho může vést ke zhoršení dalšího. Na jednotlivá kritéria mohou být kladeny různé podmínky a omezení. Takovýmito problémy se zabývá obor zvaný *Multikriteriální optimalizace* [13].

U problému, který obsahuje více kritérií pro hodnocení kvality řešení, existují kandidátní řešení, u nichž nejsme schopni rozlišit, které je kvalitnější než druhé. Tento problém je adresován tzv. *Pareto optimalitou*.

2.4.1 Pareto optimalita

Uvažujme problém, pro nějž je definována skupina funkcí

$$F = (f_1, \ldots, f_n), \tag{2.8}$$

která přiřazuje kandidátnímu řešení α hodnotu kvality $u_1, \ldots, u_n$ pro jednotlivá kritéria $1, \ldots, n$.

$$F(\alpha) = (f_1(\alpha), f_2(\alpha), \ldots, f_n(\alpha)) = (u_1, \ldots, u_n). \tag{2.9}$$

Pak α s kvalitou $F(\alpha) = (u_1, \ldots, u_n)$ *dominuje* jinému řešení β s kvalitou $F(\beta) = (v_1, \ldots, v_n)$ právě tehdy, když

$$\exists i \in \langle 1, n \rangle, u_i < v_i \wedge \forall i \in \langle 1, n \rangle, u_i \leq v_i. \tag{2.10}$$

Kandidátní řešení, které dominuje všem ostatním řešením se nazývá *Pareto–optimálním*.

Jelikož Pareto–optimálních řešení pro daný problém může být obecně více, je definována tzv. *Pareto–množina*. Pareto–množina obsahuje všechna Pareto–optimální řešení daného problému.

Pareto–frontou je pak nazývána množina PF obsahující ohodnocení $F(\alpha)$ jednotlivých kandidátních řešení α, která náleží do Pareto–množiny daného problému.

2.4.2 NSGA-II

Evolučním algoritmem uvažujícím multikriteriální optimalizaci je například algoritmus *NSGA–II* [4], který byl využit v této práci. Algoritmus funguje na obdobném principu jako ostatní evoluční algoritmy. Základním rozdílem je provádění selekce jedinců na základě stupně Pareto–dominance chromozomů a také hustoty chromozomů v okolí [4].

Stupeň Pareto–dominance je zjišťován například tímto způsobem. Nejprve je chromozomům v populaci přiřazena úroveň Pareto–dominance (kolik chromozomů z populace dominuje tomuto chromozomu). Následně je populace seřazena na základě této úrovně a stupeň Pareto–dominance je přidělen na základě pořadí [4].

Odhad hustoty populace je prováděn pro každého jedince v jednotlivých úrovních Pareto–dominance. V podstatě je pro každé kritérium problému zjišťováno, kolik chromozomů se nachází v relativní blízkosti daného jedince.

Mechanismus selekce pak preferuje ty jedince, jejichž stupeň pareto dominance je „lepší“ (jedinci dominuje méně jiných chromozomů) a kteří se nacházejí v části populace s nižší hustotou. To podporuje větší diverzitu populace a její postupnou konvergenci k co největšímu pokrytí Pareto–fronty.

Kapitola 3

Přibližné výpočty

Cílem přibližných výpočtů je snížit přesnost daného výpočtu a tím vylepšit jiné parametry s výpočtem související. Například spotřebu nebo rychlost výpočtu. Přibližné počítání využívá případu, kdy pro určité aplikace nemají relativně malé chyby vliv na celkový výsledek. Toto může být odůvodněno například nedokonalostí lidského vnímání (zrak, sluch apod.), nebo je uživatel ochoten připustit výskyt různých nepřesností, například pro prodloužení životnosti mobilní baterie. Zavádění přípustných chyb tedy může být součástí celkového návrhu systému jako další metrika, která dovoluje optimalizovat ostatní kritéria (spotřeba, celková velikost obvodu atd.) [7].

V přibližných výpočetních systémech lze zohledňování chyb zavádět na různých úrovních návrhu. Od obvodů na čipu až po algoritmus v programovacím jazyce. Příklady aplikací, u nichž je techniky přibližných výpočtů dosaženo, sahají od přibližných aritmetických obvodů (sčítačky, násobičky atd.) přes složité funkční bloky (komprese obrazu, diskrétní kosinová transformace) až po přibližné výpočetní systémy a programovací jazyky pro obecné užití [12].

3.1 Analýza citlivosti

Před samotným přistoupením k aproximaci nějakého systému je třeba vytipovat části systému, které jsou pro aproximaci vhodné. Toho je typicky dosaženo podrobnou analýzou celého systému. Postupně je testováno, jak moc jednotlivé části ovlivňují celkový výsledek systému (jeho spotřebu, zpoždění apod.). Analyzované části jsou následně buď prohlášeny za *citlivé* – tedy části, jejichž pozměňování by vedlo k nesprávnému fungování celého systému, nebo za *odolné* – tedy části, které jsou vhodné k aproximaci.

Tuto analýzu lze provádět stochasticky, kdy jsou do jednotlivých částí systému zavedeny náhodné chyby a celkový dopad na celý systém je následně vyhodnocen. Další možností může být přístup deterministický. V tomto případě je zvoleno několik stupňů aproximace, které jsou pak ručně vpravovány do jednotlivých částí systému. Podobně jako u stochastického přístupu je následně vyhodnoceno, které části jak ovlivňují celý systém [12].

3.2 Příklady chybových metrik

Důležitou součástí procesu aproximace systému je měření chyby, kterou po aproximaci systém vykazuje. Cílem je, co nejvíce aproximovat daný systém tak, aby výsledná chyba byla proti přesnému řešení co nejmenší.

Pro měření chybovosti aproximovaného systému lze využít několika technik. Většinou bývá porovnáván výstup tohoto systému se systémem přesným. Způsobů jak lze porovnání provést, je více a jsou zpravidla závislé na konkrétní aplikaci systému.

Jedním z příkladů chybové metriky pro aritmetické obvody je *maximální možná chyba* e_{wst}. Jinak ji lze také nazvat jako *nejhorší případ*:

$$e_{wst} = \max_{\forall i} |O^{(i)}_{orig} - O^{(i)}_{approx}|, \tag{3.1}$$

kdy O_{orig} představuje přesné řešení a O_{approx} řešení aproximované. Pokud je tato metrika využita jako cíl aproximace (například $e_{wst} \leq \epsilon$), je zaručeno, že výsledný systém vykazuje v nejhorším případě chybu maximálně ϵ.

Další využívanou metrikou je *maximální relativní chyba* e_{rel}:

$$e_{rel} = \max_{\forall i} \frac{|O^{(i)}_{orig} - O^{(i)}_{approx}|}{O^{(i)}_{orig}}. \tag{3.2}$$

Průměrný rozsah chyby e_{avg} je dán jako suma absolutních rozdílů přesných a aproximovaných obvodů zprůměrovaná přes n testovacích případů:

$$e_{avg} = \frac{\sum_{\forall i} |O^{(i)}_{orig} - O^{(i)}_{approx}|}{n}. \tag{3.3}$$

Chybová metrika úzce související s e_{avg} je *průměrná relativní chyba* MRE. Vyjadřuje totéž co e_{avg}, jen procentuálně:

$$MRE = \frac{e_{avg}}{maxDiff}, \tag{3.4}$$

kde $maxDiff$ je maximální možný rozdíl mezi přesnou a aproximovanou hodnotou. Porovnávají-li se například výsledné hodnoty na 8 bitech, je $maxDiff = 255$.

Pravděpodobnost chyby e_{prob} – tedy procento případů, kdy se pro danou sadu n referenčních vstupů aproximovaný výstup lišil od referenčnícho, lze vyjádřit jako:

$$e_{prob} = \frac{\sum_{\forall i} O^{(i)}_{orig} \neq O^{(i)}_{approx}}{n} \tag{3.5}$$

Jako chybovou metriku v oblasti zpracování obrazu, kterou se tato práce zabývá lze využít například *PSNR* (špičkový poměr signálu k šumu). PSNR bývá počítáno přes MSE, neboli *Mean Squared Error*, což vyjadřuje průměr kvadratických odchylek:

$$MSE = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I(i,j) - K(i,j))^2}{mn}, \tag{3.6}$$

kde m vyjadřuje počet pixelů ve sloupci obrázku, n počet pixelů v řádku, $I(i,j)$ hodnotu intenzity pixelu na souřadnicích i,j v referenčním obrázku a $K(i,j)$ hodnotu intenzity pixelu na souřadnicích i,j v obrázku posuzovaném.

Na samotné PSNR lze potom dojít přes následující vztah [11]:

$$PSNR = 20 * \log_{10}(MAX_I) - 10 * \log_{10}(MSE), \tag{3.7}$$

kde MAX_I udává maximální možnou hodnotu intenzity pixelu. Pro šedotónový obrázek $MAX_I = 255$.

Za chybovou metriku prahování lze využít metriky binárních klasifikátorů. Prahování lze považovat za binární klasifikátor rozlišující popředí obrázku od pozadí. V této oblasti se využívá kombinace dvou metrik, a to tzv. *SENSITIVITY* (citlivost) a *PRECISION* (přesnost):

$$SENSITIVITY = \frac{TP}{TP + FN}, \tag{3.8}$$

$$PRECISION = \frac{TP}{TP + FP}. \tag{3.9}$$

TP představuje tzv. *true positives*, neboli kolikrát bylo popředí správně klasifikováno jako popředí. FP pak představuje tzv. *false positives*, tedy kolikrát bylo pozadí klasifikováno jako popředí. FN, neboli *false negatives* označuje počet pixelů pozadí, které byly klasifikovány jako popředí [6].

Metriky $SENSITIVITY$ a $PRECISION$ lze zkombinovat do metriky F_{score} takto:

$$F_{score} = 2 * \frac{SENSITIVITY * PRECISION}{SENSITIVITY + PRECISION}. \tag{3.10}$$

Rozmezí hodnot F_{score} se pohybuje od 0 do 1. Dále v této práci je právě tato metrika využita k vyjádření kvality prahování. V případě $F_{score} = 1$ se jedná o bezchybné řešení. V případě $F_{score} = 0$ se pak jedná o nejhorší řešení (např. vyprahovaný obrázek je celý černý).

3.3 Aproximační techniky

Existuje více technik, kterými lze odolné části systému (analyzované, jak je popsáno v sekci 3.1) aproximovat. V této práci je využíváno přibližných sčítaček a násobiček, které spadají do kategorie aproximace digitálních obvodů. Budou proto vysvětleny principy *over-scaling* a *funkcionální aproximace* [15].

3.3.1 Voltage over-scaling

V případě této techniky je pracováno s číslicovými obvody, které byly navrženy tak, aby v určitých podmínkách vykazovaly přesné výsledky. Spotřeba energie těchto obvodů však může být snížena pomocí *voltage over-scaling* (je poskytnut zdroj energie s nižším napětím, pro které obvod vykazuje případné chybové výstupy). Tato chybovost může být způsobena např. chybami v časování. Podobně může být dosaženo zvýšení výkonu zvyšováním taktovací frekvence, což vede k podobným důvodům chybovosti obvodu. Kombinace těchto dvou přístupů je využívána v přístupu *dynamic over-scaling* [15].

3.3.2 Funkcionální aproximace

V této technice je navrhován obvod, který vykazuje specifikované chování jen zčásti. To tak, že obvod implementuje podobnou (ale jinou) funkci, která vykazuje přijatelnou chybovost a spotřeba energie či jiné parametry jsou adekvátně optimalizovány [12].

Důležité je u této techniky měření chybovosti navrženého obvodu. Možnosti tohoto měření byly popsány v sekci 3.2.

Jednoduchým přístupem může být v aritmetických obvodech například odseknutí nejméně významných bitů. Dalším způsobem může být odpojování jednotlivých komponent obvodu, či jejich přeskládání tak, aby obvod vykazoval nižší zpoždění.

Dalším přístupem (využitým i v této práci) je nahrazování přesných komponent obvodu komponentami již aproximovanými.

Obecně lze samotné provádění těchto metod rozdělit na dvě kategorie.

Ruční přístup

U tohoto přístupu navrhuje aproximaci obvodu designér, který má zkušenosti z danou problematikou. Na základě těchto zkušeností vybírá části systému k aproximaci. Výsledný obvod je následně vyhodnocen pomocí předem zvolených metrik a omezení. Neodpovídá-li řešení požadovaným podmínkám, musí designér navrhovat znovu. Tento přístup je evidentně složitý. Především tehdy, kdy je možných řešení mnoho a tvoří rozsáhlý stavový prostor, který není možno v přijatelném čase prohledat celý. V současné době se proto přechází k technikám, které tento postup automatizují.

Automatický návrh

Problém aproximace je v tomto případě formalizován a předložen odpovídajícím algoritmům. Je definován postup vytvoření daných aproximací a vyhodnocení jejich kvality. Je žádoucí, aby algoritmus postupně konvergoval ke kvalitnějším řešením. V této práci byly využity právě evoluční algoritmy popsané v kapitole 2.

Kapitola 4

Genetické vylepšování a návrh algoritmů pomocí přibližných výpočtů

Tato práce se, jak již bylo zmíněno, zabývá využitím přibližných výpočtů v oblasti zpracování obrazu. Jako způsob aproximace byla vybrána technika funkcionální aproximace (sekce 3.3.2). Samotný návrh aproximovaného obvodu je automatizován (úplně, případně částečně). Automatizaci zajišťují evoluční algoritmy (kapitola 2). Konkrétně pak ES v kombinaci s CGP a NSGA-II. Aproximace obvodu je prováděna nahrazováním přesných součástek již navrženými aproximovanými součástkami. Tyto součástky byly převzaty z knihovny *EvoApproxLib*[1] [9]. Konkrétně byly využity aproximované sčítačky a násobičky.

Aplikační oblastí, na kterou byly techniky užity je *adaptivní prahování psaného textu*. V této kapitole jsou popsány dva přístupy, pomocí nichž jsou tyto techniky aplikovány. Jedním z nich je vylepšování softwaru pomocí aproximace částí existujícího algoritmu (sekce 4.2). Druhým pak samotný návrh algoritmu po poskytnutí aproximovaných součástek (sekce 4.3).

U obou přístupů byla využita sada obrázků textu a jim příslušející referenční vyprahované obrázky. Tato sada byla převzata z datasetu *DIBCO2009*[2] („Document image binarization contest“) [10].

4.1 Aproximované sčítačky a násobičky

V tabulkách 4.1 a 4.2 jsou vypsány parametry jednotlivých sčítaček a násobiček, jež byly vybrány z *EvoApproxLib*. Součástky byly vybírány z množiny osmibitových bezznaménkových sčítaček a násobiček. Vybrané součástky patří do Pareto–optimální podmnožiny vzhledem k metrice MAE (sekce 3.2) a spotřebě.

Dále v této práci bude k vyjádření stupně aproximace užitých součástek využívána odpovídající metrika MRE (popsána v sekci 3.2).

[1] https://ehw.fit.vutbr.cz/evoapproxlib/

[2] https://users.iit.demokritos.gr/ bgat/DIBCO2009/benchmark/

Název sčítačky	MAE	WCE	MRE	EP	Spotřeba	Plocha
add8u_0FP	0.00 %	0.00 %	0.00 %	0.00 %	0.033 mW	70.4 μm^2
add8u_5R3	0.04 %	0.20 %	0.14 %	25.00 %	0.029 mW	63.8 μm^2
add8u_5QL	0.16 %	0.59 %	0.40 %	43.75 %	0.024 mW	57.3 μm^2
add8u_5LT	0.33 %	1.37 %	0.91 %	71.88 %	0.021 mW	56.3 μm^2
add8u_5HQ	0.68 %	2.93 %	1.80 %	85.74 %	0.017 mW	53.0 μm^2
add8u_5SY	1.05 %	3.12 %	2.93 %	94.14 %	0.012 mW	28.2 μm^2
add8u_099	2.03 %	6.64 %	6.23 %	97.07 %	0.0095 mW	25.8 μm^2
add8u_006	4.92 %	17.97 %	14.58 %	98.77 %	0.0046 mW	15.0 μm^2
add8u_08V	9.88 %	30.47 %	24.87 %	99.45 %	0.0015 mW	8.0 μm^2
add8u_063	15.29 %	47.66 %	37.63 %	99.61 %	0.000 mW	0.0 μm^2

Tabulka 4.1: Aproximované sčítačky

Název násobičky	MAE	WCE	MRE	EP	Spotřeba	Plocha
mul8u_1JFF	0.000 %	0.000 %	0.00 %	0.00 %	0.391 mW	709.6 μm^2
mul8u_EXZ	0.001 %	0.015 %	0.033 %	19.53 %	0.380 mW	663.6 μm^2
mul8u_150Q	0.008 %	0.064 %	0.15 %	37.30 %	0.360 mW	660.3 μm^2
mul8u_2AC	0.037 %	0.12 %	1.25 %	98.12 %	0.311 mW	508.3 μm^2
mul8u_185Q	0.18 %	0.79 %	4.16 %	98.05 %	0.206 mW	427.5 μm^2
mul8u_FTA	0.89 %	4.29 %	13.96 %	98.74 %	0.084 mW	214.5 μm^2
mul8u_13QR	4.83 %	19.46 %	44.00 %	99.20 %	0.0085 mW	41.3 μm^2
mul8u_199Z	24.81 %	99.22 %	100.00 %	99.22 %	0.00 mW	0.0 μm^2

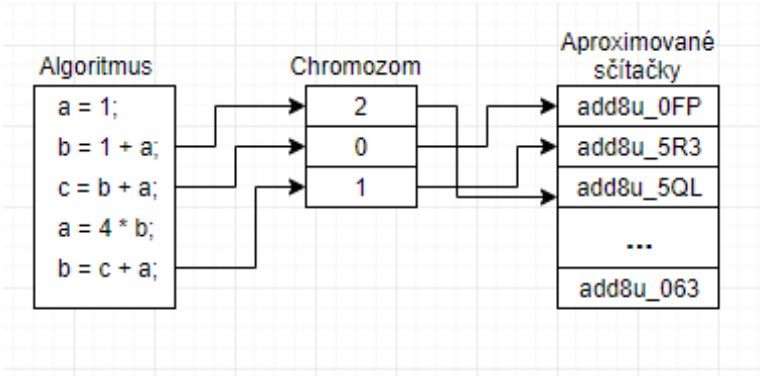
Tabulka 4.2: Aproximované násobičky

4.2 Aproximace existujícího algoritmu

Jako první technika byla zvolena aproximace existujícího algoritmu. Princip této techniky funguje následovně. Je poskytnut existující algoritmus. V tomto algoritmu jsou detekovány části kódu, které mají být aproximovány. Těmito částmi kódu jsou v tomto případě operace sčítání. Následně dochází k aproximaci vybraných částí. Aproximace je provedena nahrazením přesného sčítání aproximovanými sčítačkami z *EvoApproxLib*. Existuje mnoho způsobů, jak přesnou sčítačku nahradit. Obecně je snahou, aby byl algoritmus aproximován co nejvíce (vyšší aproximace předpokládá levnější řešení). Zároveň aby zanesená chyba byla co nejnižší. Jelikož je v tomto případě třeba sledovat dvě kritéria, byl pro nalezení pareto-optimálních řešení použit algoritmus NSGA-II. Implementace algoritmu NSGA-II, byla převzata z [4] a pro vlastní potřeby upravena. Způsob reprezentace chromozomu, získání fitness a trénovací podmnožiny datasetu je popsán níže.

4.2.1 Chromozom

Zakódování kandidátního řešení do chromozomu lze pozorovat na obrázku 4.1. Geny chromozomu reprezentují jednotlivá sčítání v aproximovaném algoritmu. Samotná hodnota genu pak udává, která varianta aproximované sčítačky se má použít namísto odpovídajícího přesného součtu.



Obrázek 4.1: Reprezentace chromozomu v užitém algoritmu. Geny chromozomu reprezentují jednotlivá sčítání v algoritmu. Hodnota genu definuje užitou aproximovanou sčítačku.

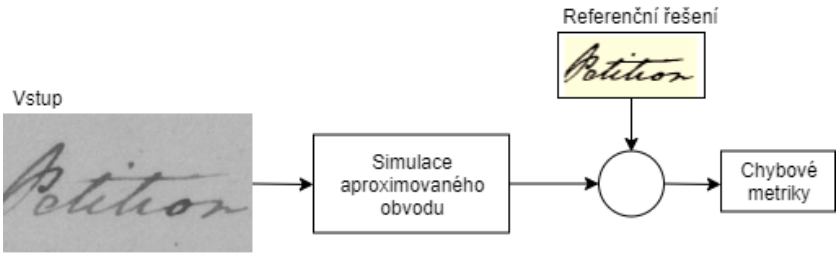
4.2.2 Získání fitness

Zjišťování chybových metrik jedinců v populaci lze pozorovat na obrázku 4.2. Na vstup aproximovaného algoritmu, jenž je reprezentován chromozomem, je přiveden trénovací obrázek. Algoritmus je pak simulován a jeho výstup je porovnáván s referenčním řešením.

Po skončení evoluce je pro všechny jedince výsledné Pareto–fronty vypočítána fitness znovu. Tentokrát je však jedinec simulován pro celý dataset. Tyto hodnoty jsou pak srovnávány s kvalitou neaproximovaného řešení.

Zvolenou chybovou metrikou je pro tento případ metrika F_{score}, popsaná v sekci 3.2. Tato metrika je normalizovaná do intervalu $\langle 0, 1 \rangle$. Jelikož bylo k problému přistoupeno jako k minimalizaci, odpovídá první kritérium evolučního algoritmu součinu $-1 * F_{score}$.

Aby algoritmus preferoval vyšší stupeň aproximace v rámci stejné kvality řešení, bylo přidáno kritérium popisující celkový stupeň aproximace obvodu. Toto kritérium odpovídá celkovému součtu hodnot MRE užitých součástek. Aby docházelo k minimalizaci, je tento součet opět negován.



Obrázek 4.2: Zjišťování chybových metrik v užitém algoritmu. Aproximovaný algoritmus je simulován na vstupním obrázku. Výstup tohoto algoritmu je pak porovnáván s referenčním řešením.

4.2.3 Genetické operátory

Chromozom obsahuje indexy, které zastupují jednotlivé aproximované sčítačky. Při mutaci genu je tedy gen změněn na jiný index v daném rozsahu. Tato změna je prováděna jako náhodné vygenerování celého čísla v rozsahu $\langle 0, n \rangle$, kdy n značí celkový počet poskytnutých sčítaček. V případě vygenerování indexu, který odpovídá hodnotě genu před mutací je generování opakováno.

Křížení je pak provedeno jako průměr hodnot dvou rodičovských genů zaokrouhlený dolů. Jelikož jsou ze dvou rodičů generováni dva potomci, je druhému potomkovi gen inkrementován. Pokud by došlo k hodnotě genu mimo rozsah, k inkrementaci nedojde.

4.2.4 Dataset vs trénovací podmnožina

Samotnému evolučnímu procesu není poskytnut celý dataset, ale pouze jeho podmnožina. To především kvůli úspoře času, po který evoluce poběží. Poskytnutí příliš malé podmnožiny však může vést k problémům. Fitness jedince na trénovací množině pak nekoreluje s fitness pro celý dataset. V experimentech s NSGA-II byla velikost trénovací podmnožiny experimentálně zvolena tak, aby se rozdíl fitness ve většině případů nelišil o více než 20%.

4.2.5 Hypotéza

Předpokladem je, že vyšší aproximace přináší levnější řešení. Předpokládá se také, že aproximované řešení bude vzhledem k fitness horší, než řešení přesné. Cílem experimentů bude zjistit, zda ke zhoršení dojde, případně v jaké míře.

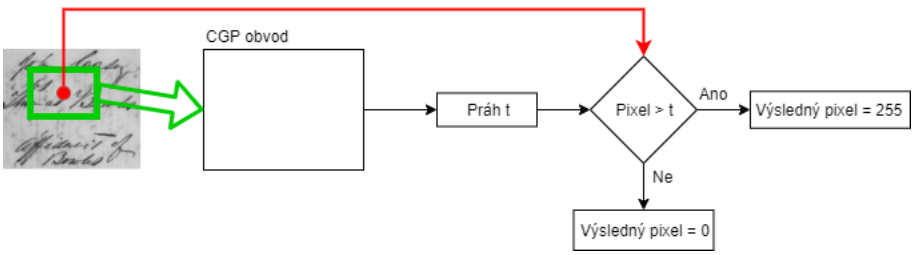
Samotný algoritmus, jenž je dále aproximován, byl převzat z [3]. Jedná se o algoritmus, který pro výpočet prahu využívá integrální obraz.

4.3 Navržení aproximovaného řešení pomocí CGP

Dalším přístupem pro návrh aproximovaného obvodu je plně automatizovaný návrh pomocí CGP v kombinaci s ES. V této práci je využit podobný přístup jako při návrhu konvolučních filtrů v [14]. To proto, že i adaptivní prahování pracuje s okolím pixelu, aby byla zjištěna hodnota prahu pro daný pixel. Chybovou metrikou prahování je podobně jako v předchozí sekci 4.2 zvolena metrika F_{score}.

4.3.1 Vstup a výstup CGP

Algoritmy implementující adaptivní prahování obrázku počítají práh z okolí prahovaného pixelu. Tímto postupem je inspirován způsob reprezentace CGP mřížky. Obvod, který CGP reprezentuje, odpovídá výpočtu prahu z okolí pixelu P. Okolí prahovaného pixelu $O(P)$ je tedy přivedeno na vstupy CGP. Výstup CGP odpovídá hodnotě prahu t. Prahovaný pixel je pak prahován s tímto prahem. Schéma lze pozorovat na obrázku 4.3.



Obrázek 4.3: Postup simulace aproximovaného obvodu pro jeden pixel testovacího obrázku. Zelený obdélník odpovídá okolí prahovaného pixelu, jež je přivedeno CGP na vstup. Červeně zvýrazněný pixel je pak prahován s výstupem CGP.

4.3.2 Výpočet fitness

Pro výpočet fitness je využita metrika F_{score} popsaná v sekci 3.2. Tato metrika je přímo považována za optimalizované kritérium. Jde tedy o problém maximalizace. Dalším sledovaným kritériem je celková plocha $AREA$ uzlů CGP, jež jsou pro výpočet využity. Předpokládejme, že $AREA(i)$ definuje plochu uzlu i. Zároveň předpokládejme, že $used(i) = 1$, pokud je uzel při výpočtu využit a $used(i) = 0$ v případě nevyužití uzlu. Pak lze $AREA$ definovat jako

$$AREA = \sum_{\forall i} AREA(i), i \in \langle 0, n \rangle \wedge used(i) = 1, \tag{4.1}$$

kdy n značí celkový počet uzlů v mřížce.

ES optimalizuje pouze jedno kritérium. Výběr nové populace je proto upraven následovně. Přednost dostávají jedinci s vyšší fitness (metrika F_{score}). V případě rovnosti F_{score} je dána přednost jedinci s nižší celkovou plochou $AREA$.

Z důvodu úspory délky běhu není evoluci poskytnut celý dataset. Je vybrána pouze podmnožina vstupních pixelů, jejich okolí a odpovídající referenční výstup. Výběr této podmnožiny je prováděn náhodně. Zároveň je však dbáno na to, aby byl rovnoměrný poměr zastoupení případů, kdy je referenční výstup $out_{ref} = 0$ a $out_{ref} = 255$. Může nastat, že jsou již vybrány všechny případy, kdy $out_{ref} = 0$. Potom jsou již vybírány pouze případy, kdy $out_{ref} = 255$. Počet bílých pixelů je totiž u psaného textu vyšší. Výběr je prováděn tak, aby se v trénovací množině nenacházely duplikáty.

Pro nejlepšího jedince, který je výsledkem evoluce, je pak vypočítána fitness z celého datasetu. Tato fitness je pak uvažována pro statistické srovnávání.

4.3.3 Sada funkcí

Základem sady funkcí, která je poskytnuta CGP, jsou aproximované sčítačky a násobičky z EvoApproxLib. Plocha součástek je u těchto již uvedena. Jelikož však evoluce pouze s těmito funkcemi nenacházela řešení, které by bylo dostatečně kvalitní, byla ještě přidána funkce bitového posunu doprava:

$$out = in_1 >> 1. \tag{4.2}$$

Zde out značí výstup uzlu a in_1 první ze vstupů uzlu. Plocha $AREA(shift)$ tohoto uzlu je uvažována jako $AREA(shift) = 0$ a je tedy při výpočtu celkové plochy obvodu zanedbána. To proto, že této funkce lze dosáhnout pouhým posunutím propoje mezi jednotlivými součástkami o jeden bit.

4.3.4 Genetické operátory

Evolučnímu algoritmu ES je poskytnut pouze jeden genetický operátor. Křížení v tomto případě není využito. V každé generaci jsou potomci vygenerováni pouze mutací jediného chromozomu (strategie $1 + \lambda$). Samotná mutace je pak prováděna změnou náhodného genu na náhodnou hodnotu z platného rozsahu. V každé generaci je potomek vygenerován náhodnou změnou n genů rodiče.

Kapitola 5

Návrh experimentů

V této kapitole jsou popsány experimenty, pomocí nichž bude testován vliv aproximace na kvalitu prahování psaného textu. U obou metod, které byly popsány v kapitole 4, jsou nejprve popsány experimenty poskytující podklad k vhodnému nastavení parametrů evoluce. Následně jsou popsány samotné experimenty, jež zkoumají, jak se projeví aproximace na výsledcích prahování.

5.1 Aproximace existujícího algoritmu

V této sekci jsou popsány experimenty související s aproximací existujícího algoritmu adaptivního prahování. Nejprve jsou popsány experimenty poskytující podklad pro nastavení parametrů evolučního algoritmu NSGA-II. Následně pak experiment, který testuje vliv aproximace na kvalitu prahování.

Pro tyto experimenty slouží program `nsga2`, jehož základní implementace byla převzata[1] a upravena pro potřeby adaptivního prahování textu a využití aproximovaných sčítaček.

5.1.1 Celkový počet generací

První navrhnutý experiment poskytuje podklad pro zvolení vhodného celkového počtu generací. Evolučnímu algoritmu bude poskytnuta trénovací podmnožina datasetu o velikosti 50% celkové velikosti datasetu. Velikost populace bude 52 (implementace vyžduje dělitelnost čtyřmi). Pravděpodobnost křížení bude nastavena na $P_{cross} = 80\,\%$ a pravděpodobnost mutace na $P_{mutation} = 50\,\%$. Celkový počet generací bude postupně zvyšován. Následně budou porovnávány výsledné Pareto–fronty běhů různých konfigurací. Jakmile se výsledná Pareto–fronta při dalším zvyšování počtu generací nebude výrazně zlepšovat, bude tento počet generací zvolen za dostačující pro další experimenty.

5.1.2 Velikost trénovací podmnožiny datasetu

Zjištění vhodné trénovací podmnožiny datasetu bude prováděno následovně. Z výsledků předchozího experimentu bude zvolen vhodný počet generací. Pravděpodobnost křížení bude nastavena na $P_{cross} = 80\,\%$ a pravděpodobnost mutace na $P_{mutation} = 50\,\%$. Velikost populace bude 52. Evoluci bude poskytována trénovací podmnožina datasetu o různé velikosti. Postupně se bude tato podmnožina zvětšovat. Po skončení evoluce bude pro nedominovaná řešení zjištěna fitness pro celý dataset. Následně bude spočítán rozdíl této fitness

[1] https://www.iitk.ac.in/kangal/codes.shtml

oproti fitness trénovací. Pro další experimenty pak bude jako vhodná konfigurace trénovací podmnožiny zvolena ta, kdy chyba trénovací fitness ve většině případů nepřekročí 20%.

5.1.3 Velikost populace

Tento experiment poskytuje podlad pro zvolení vhodné velikosti populace chromozomů v jedné generaci. Z předchozích experimentů již bude zvolena vhodná velikost trénovací podmnožiny datasetu a celkový počet generací evoluce. Pravděpodobnost křížení bude nastavena na $P_{cross} = 80\,\%$ a pravděpodobnost mutace na $P_{mutation} = 50\,\%$. Budou poskytnuty velikosti populace 52, 104, 208. Aby nedošlo k většímu prohledávání prostoru, bude zároveň počet generací upraven tak, aby $\lambda * Count_{generations} = C$, kdy C je konstantní.

5.1.4 Zkoumání vlivu aproximace

Z předchozích experimentů této podsekce je vytvořen základ pro experimenty zjišťující dopad aproximace na celkovou kvalitu prahování. Evoluci budou poskytnuty aproximované sčítačky z *EvoApproxLib*, jejichž parametry jsou uvedeny v tabulce 4.1. Konfigurace jednotlivých parametrů evoluce budou zvoleny na základě předchozích experimentů. Pravděpodobnost křížení bude nastavena na $P_{cross} = 80\,\%$ a pravděpodobnost mutace na $P_{mutation} = 50\,\%$. Po doběhnutí experimentů budou zkoumány výsledné Pareto–fronty a rozdíl výsledků součástek s vyšším stupněm aproximace oproti součástkám přesným.

5.2 Navržení aproximovaného řešení pomocí CGP

V této sekci jsou popsány experimenty související s návrhem aproximovaného řešení pomocí CGP. Nejprve jsou popsány experimenty poskytující podklady pro nastavení vhodných parametrů evoluce. Následně je popsáno experimentování, jehož cílem je zkoumat vliv aproximace na kvalitu nalezených řešení. Parametr l_{back} bude ve všech experimentech ponechán na své maximální hodnotě.

Pro tyto experimenty slouží program `cgp`, jehož základní implementace byla převzata[2] a upravena pro potřeby adaptivního prahování a využití aproximovaných součástek.

5.2.1 Celkový počet generací

V prvním experimentu této sekce je zkoumán potřebný počet generací pro nalezení dostatečně kvalitních řešení. Postupně bude zvyšován poskytnutý počet generací a následně bude zkoumán vývoj fitness nejlepších jedinců, kteří byli v nezávislých bězích nalezeni. Číslo generace, po které se fitness většiny jedinců do konce evoluce nezmění, bude prohlášeno jako dostatečný počet generací. Je pravděpodobné, že někteří jedinci se budou zlepšovat i po této generaci. Budou však zanedbáni.

5.2.2 Velikost trénovací podmnožiny datasetu

Zjištění vhodné velikosti trénovací podmnožiny datasetu bude prováděno následovně. Počet trénovacích vektorů bude postupně zvyšován. Při neposkytnutí celého datasetu se zřejmě bude fitness jedinců lišit od fitness spočítané přes celý dataset. Jakmile budou jedinci dosahovat fitness, jež se bude lišit o méně než 20 %, bude odpovídající velikost trénovací podmnožiny datasetu uvažována jako vhodná pro další experimenty.

[2] http://www.fit.vutbr.cz/∼vasicek/cgp/

5.2.3 Velikost okolí pixelu

Vhodná velikost okolí pixelu bude zjišťována následujícím experimentováním. Pro nezávislé experimenty bude poskytnuto okolí pixelu 3x3, 5x5 a 7x7. Ostatní parametry budou vzaty z výsledků předchozích experimentů. Po nalezení nejlepších jedinců, bude pro tyto vypočítána fitness pro celý dataset. Následně budou porovnány výsledky pro různou velikost okolí. Vhodnou velikostí okolí pak bude prohlášeno to, které bude nalézat nejkvalitnější jedince. V případě obdobných výsledků mezi jednotlivými konfiguracemi bude preferováno okolí menší.

5.2.4 Zkoumání vlivu aproximace

Z předchozích experimentů této sekce budou převzaty vhodné parametry evoluce. Následně bude zkoumán vliv poskytnutí součástek s různým stupněm aproximace na výsledné kvalitě prahování. Evoluci budou v experimentech poskytnuty různé kombinace jedné aproximované sčítačky a jedné aproximované násobičky. Množina aproximovaných součástek, ze kterých budou kombinace tvořeny, lze pozorovat v tabulce 4.1 a 4.2. Pro všechny tyto kombinace bude navíc experimentováno s různým tvarem CGP mřížky. Konkrétně s mřížkami 1x10, 5x10 a 10x10. Následně budou porovnávány výsledky jednotlivých konfigurací. Bude zkoumána dosažená kvalita prahování nejlepších jedinců. Dále bude zkoumána celková plocha navržených obvodů.

Kapitola 6

Experimentální výsledky

Bylo prováděno několik sad experimentů, kterými bylo zkoumáno, jaký vliv má využití přibližných výpočtů na kvalitu adaptivního prahování psaného textu. Nejprve z hlediska F_{score}, kde se intuitivně předpokládá zhoršení výsledků po aproximaci (nicméně nemusí tak tomu nutně být). U návrhu pomocí CGP + ES je zkoumána také plocha navrženého obvodu.

Experimenty byly prováděny na superpočítači *Anselm*[1]. Pro každé nastavení parametrů evoluce bylo spuštěno 48 nezávislých běhů. Výsledky těchto běhů byly následně agregovány a vyneseny do grafu.

Zavádění aproximací bylo aplikováno na oblast ze zpracování obrazu. Konkrétně na téma adaptivního prahování psaného textu. Jako testovací množina byly využity obrázky psaného textu a jejich odpovídající referenční „*ground truth*“ výsledky po aplikaci prahování. Tyto obrázky byly převzaty z [10].

Evolučnímu algoritmu byly poskytnuty již aproximované součástky z *EvoApproxLib*.

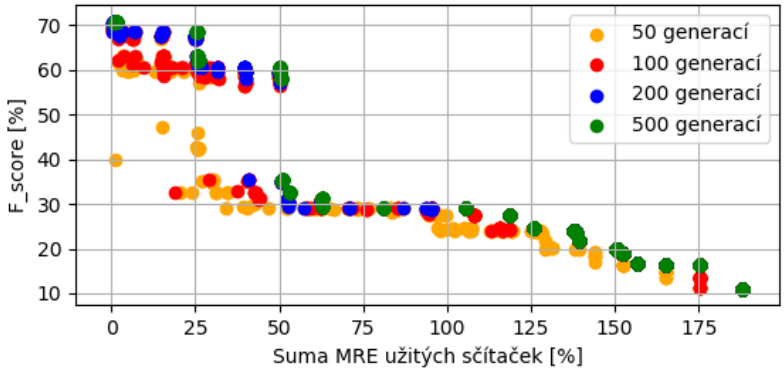
6.1 Aproximace existujícího algoritmu

V této sekci budou rozebrány výsledky experimentů souvisejících s aproximací existujícího algoritmu. Nejprve budou rozebrány výsledky experimentů, které poskytují podklady pro vhodné nastavení parametrů evoluce. Následně bude zkoumán vliv aproximace na kvalitu nalezených řešení vzhledem ke kvalitě prahování.

6.1.1 Celkový počet generací

Jednotlivé konfigurace tohoto experimentu byly popsány v sekci 5.1.1. Výsledné Paretofronty všech jednotlivých běhů byly vykresleny do grafu (nejde tedy o celkovou Paretofrontu v rámci všech nezávislých běhů). Výsledky lze pozorovat na obrázku 6.1. Při zvyšování počtu generací se výsledky zlepšují. Nicméně mezi výsledky generace 200 a generace 500 nedošlo k zásadnímu zlepšení. Pro další experimenty proto bude pro populaci 52 jedinců uvažován počet generací 200 jako dostatečný.

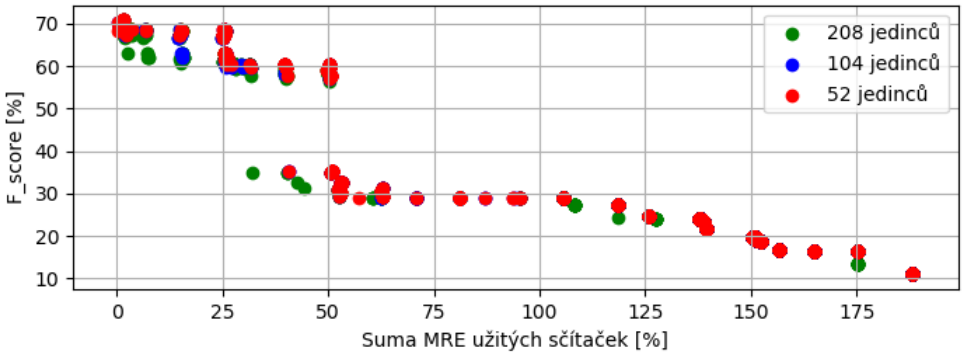
[1] https://docs.it4i.cz/anselm/introduction/



Obrázek 6.1: Vykreslení Pareto–front všech nezávislých běhů konfigurací s různým počtem generací. Kandidátní řešení s vyšší F_{score} a vyšší sumou MRE užitých sčítaček je považováno za lepší. Výsledky pro konfiguraci s 500 generacemi (zeleně) nejsou zásadně lepší než výsledky konfigurace s 200 generacemi (modře).

6.1.2 Velikost populace

Pro jednotlivé konfigurace popsané v sekci 5.1.3 bylo prováděno 48 nezávislých běhů. Pareto-fronty těchto běhů pak byly pro jednotlivé konfigurace evoluce vykresleny do grafu, který lze pozorovat na obrázku 6.2. Výsledky jednotlivých konfigurací vychází podobně pro všechny varianty. Konfigurace s 208 jedinci v populaci však vychází lehce hůře. Pro další experimentování bude pracováno s velikostí populace 52 jedinců.



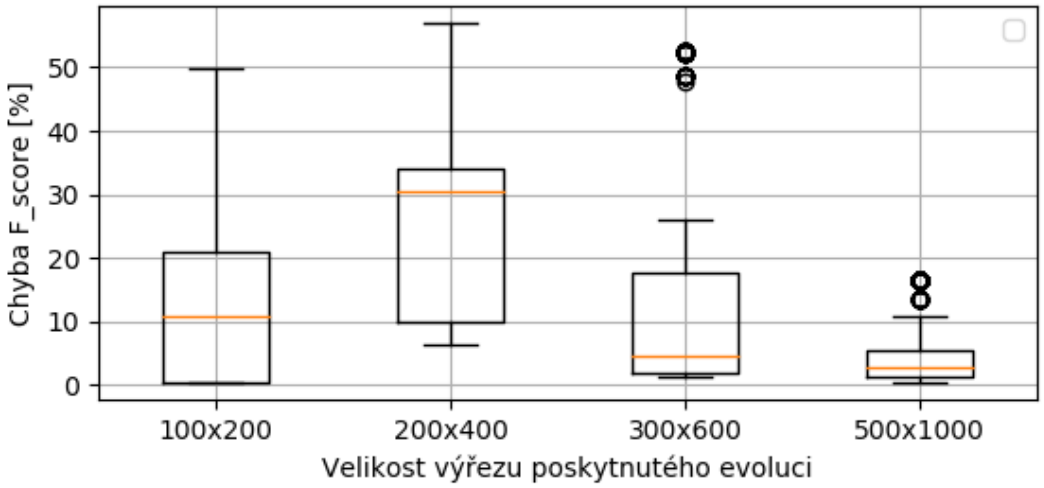
Obrázek 6.2: Vykreslení Pareto—front všech nezávislých běhů konfigurací s různou velikostí populace. Kandidátní řešení s vyšší F_{score} a vyšší sumou MRE užitých sčítaček je považováno za lepší. Výsledky pro jednotlivé konfigurace nejsou zásadně odlišné. Konfigurace s 208 jedinci v populaci (zelená) však vykazuje většinu nejhorších jedinců.

6.1.3 Velikost trénovací podmnožiny

V tomto experimentu bylo zkoumáno, jaká velikost trénovací podmnožiny datasetu je potřebná pro poskytnutí odpovídající fitness jedinců. Evolučnímu algoritmu byl poskytnut

výřez obrázku z datasetu. Výsledné Pareto–frontě nalezených řešení byla vypočítána fitness pro celý obrázek. Následně byla zkoumána velikost rozdílu těchto dvou fitness.

Výsledky experimentu lze pozorovat na obrázku 6.3. Původní rozměry poskytnutého obrázku jsou 581x1091 pixelů. Jak je vidět z výsledků, poskytování menších výřezů z obrázku produkuje výraznou chybu fitness. Přijatelné výsledky jsou až u výřezu o rozměrech 500x1000 pixelů. To může být způsobeno například tím, že užitý algoritmus počítá s okolím pixelu o šířce, která je odvozena ze šířky okna. Po poskytnutí příliš malého výřezu je tedy počítáno s menším okolím. Užité sčítačky jsou použity méněkrát. Chyba aproximace je tedy ve výsledku nižší, než při výpočtu s okolím větším. V dalších experimentech bude proto pracováno s výřezem, který odpovídá výřezu 500x1000 pixelů u obrázku *H04* (viz příloha B).



Obrázek 6.3: Závislost chyby F_{score} počítané pro trénovací podmnožinu datasetu na velikosti poskytnutého výřezu obrázků. Zde uváděny velikosti výřezu pro obrázek *H04* (viz příloha B).

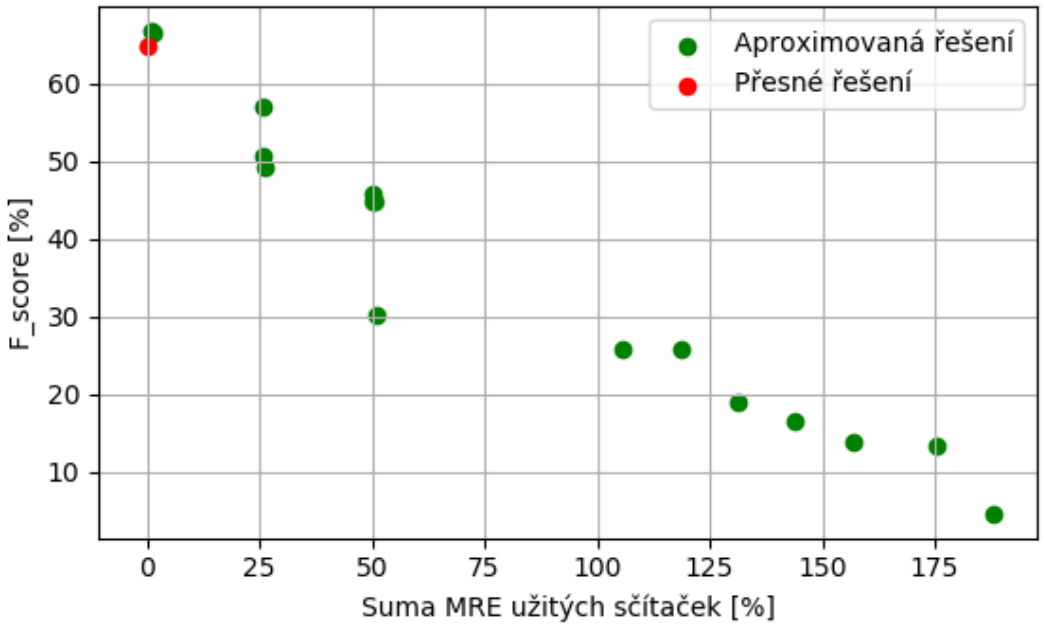
6.1.4 Vliv aproximace na kvalitu prahování

Z předchozích experimentů byly převzaty vhodné parametry pro evoluční algoritmus. Následně byly spuštěny nezávislé běhy s těmito parametry. Výslednou Pareto-frontu všech nejlepších nalezených řešení lze pozorovat na obrázku 6.4.

Z výsledků lze pozorovat, že aproximovaná řešení vycházejí většinou podstatně hůře, než řešení přesné. Pouhá dvě aproximovaná řešení jsou jemně lepší, než řešení přesné. Aproximace těchto řešení je však minimální. Při vyšší aproximaci lze pozorovat významný pokles kvality prahování (viz nalezené řešení 3) v tabulce 6.1).

Na obrázku 6.5 lze pozorovat rozdíly prahování jednotlivých nalezených řešení, přesného řešení a referenčního výstupu. S vyšší mírou užití aproximovaných součástek lze skutečně pozorovat pokles kvality prahování.

Z výsledků těchto experimentů plyne, že se tento způsob zavádění aproximací nevyplatí. Otázkou je, jaký má vliv aproximace na další parametry, například celkovou plochu obvodu. Těmito parametry se budou zabývat experimenty popsané v sekci 5.2.



Obrázek 6.4: Výsledná Pareto–fronta nejlepších řešení v rámci 48 nezávislých běhů evolučního algoritmu. Zobrazena závislost kvality prahování vyjádřené metrikou F_{score} na stupni aproximace algoritmu vyjádřeném sumou MRE všech užitých sčítaček.

Nalezené řešení	F_{score}	**Suma MRE**
1)	66.7625 %	1.2 %
2)	66.5494 %	1.6 %
3)	56.9604 %	25.81 %
Přesné řešení	64.7832 %	0 %

Tabulka 6.1: Tabulka s výsledky nejlepších řešení. Očíslování nalezených řešení odpovídá pořadí Řešení ve výsledné Pareto–frontě, kterou lze pozorovat na obrázku 6.4. Řazeno od nejlepší F_{score} k nejhorší.

Vstupní obrázek | Referenční výstup

Přesné řešení | Aproximované řešení 1)

Aproximované řešení 2) | Aproximované řešení 3)

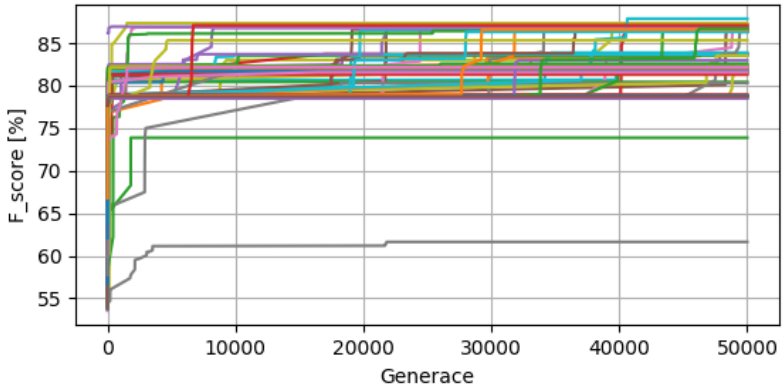
Obrázek 6.5: Výstupy prahování nalezených aproximací užitého algoritmu. Kandidátní řešení odpovídají uvedeným v tabulce 6.1. Pro zvýraznění rozdílů je obrázek zvětšen a oříznut.

6.2 Návrh pomocí CGP

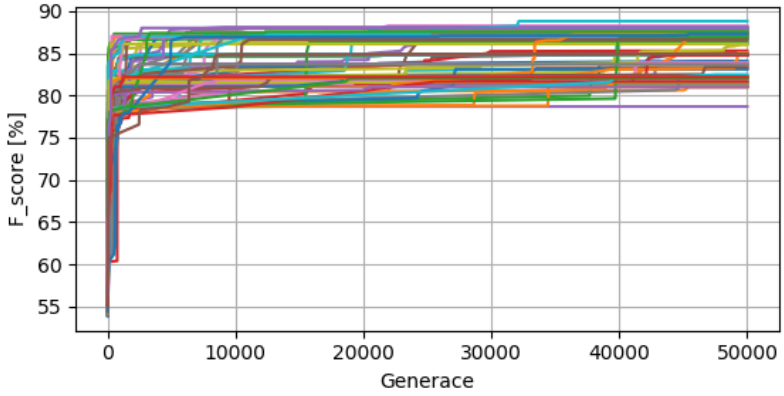
V této sekci jsou rozebrány výsledky experimentů souvisejících s návrhem obvodu pomocí CGP a ES. Nejprve jsou rozebrány výsledky experimentů, které poskytují podklady pro vhodné nastavení parametrů evoluce. Následně je zkoumán vliv aproximace na kvalitu nalezených řešení vzhledem ke kvalitě prahování a celkové ploše odpovídajících obvodů. Veškeré experimenty byly prováděny pro různý tvar mřížky CGP. Počet sloupců n zůstává stejný, a to $n = 10$. Počet řádků je vybírán ze 3 variant $m = 1$, $m = 5$ a $m = 10$ řádků. Parametr l_{back} je nastaven na maximální možnou hodnotu $l_{back} = 10$. U všech experimentů je prováděno 48 nezávislých běhů.

6.2.1 Celkový počet generací

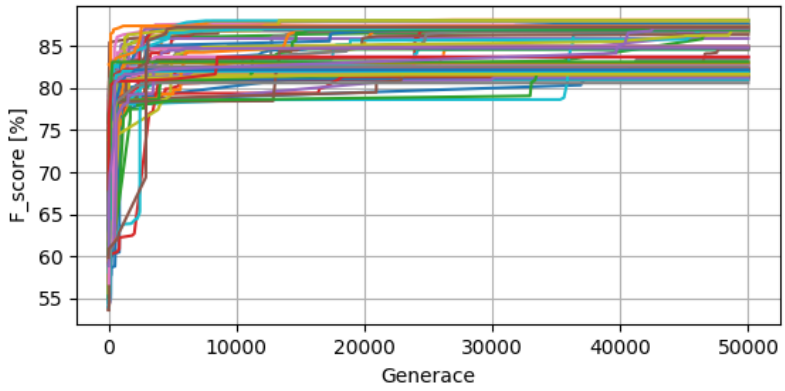
Vývoj fitness jedinců pro různý tvar mřížky lze pozorovat na obrázcích 6.6, 6.7 a 6.8. U všech tří variant lze pozorovat největší zlepšení fitness do generace $g_1 = 10\,000$. Následně dochází k pomalejšímu zlepšování. Po generaci $g_2 = 30\,000$ se již naprostá většina jedinců nezlepší. Pro další experimenty bude proto uvažován jako dostatečný počet generací $g_{count} = 30\,000$.



Obrázek 6.6: Vývoj fitness nejlepších jedinců v závislosti na počtu generací. Zobrazen záznam 48 nezávislých běhů pro tvar mřížky CGP 1x10.



Obrázek 6.7: Vývoj fitness nejlepších jedinců v závislosti na počtu generací. Zobrazen záznam 48 nezávislých běhů pro tvar mřížky CGP 5x10.



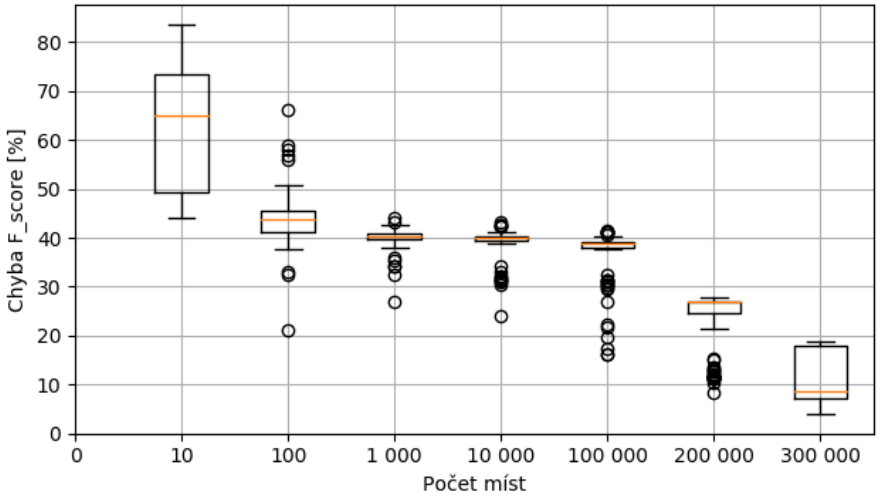
Obrázek 6.8: Vývoj fitness nejlepších jedinců v závislosti na počtu generací. Zobrazen záznam 48 nezávislých běhů pro tvar mřížky CGP 10x10.

6.2.2 Velikost trénovací podmnožiny

Na obrázku 6.9 lze pozorovat závislost chyby F_{score} na velikosti trénovací podmnožiny datasetu, kerá byla evoluci poskytnuta.

Z důvodu velké časové náročnosti pro celý dataset byl celkový dataset omezen pouze na obrázek H04 (viz příloha B). Trénovací podmnožina pak obsahuje místa náhodně vybraná z tohoto obrázku.

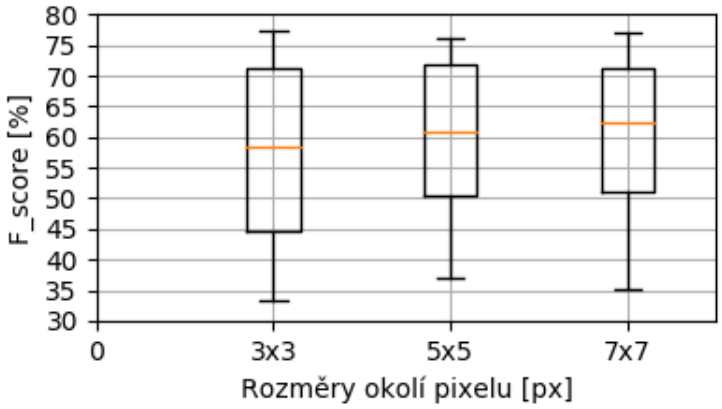
Chyba F_{score} se s narůstajícím počtem poskytnutých míst snižuje. Výrazněji to však lze pozorovat až od počtu 200 000 poskytnutých míst. V tomto případě je již dostatek případů, kdy F_{score} pro trénovací množinu dostatečně odpovídá F_{score} pro celý dataset. Velikost trénovací podmnožiny v tomto případě odpovídá zhruba 15 % velikosti celého datasetu. U dalších experimentů je proto evoluci poskytnuta podmnožina datasetu právě o této velikosti.



Obrázek 6.9: Chyba F_{score} v závislosti na velikosti trénovací podmnožiny datasetu. Chyba F_{score} je počítána jako absolutní rozdíl F_{score} počítané pro celý dataset a F_{score} počítané pro trénovací podmnožinu datasetu.

6.2.3 Velikost okolí pixelu

Na obrázku 6.10 lze pozorovat, jakým způsobem se projeví různá velikost okolí prahovaného pixelu na výsledné F_{score}. Výsledky jednotlivých konfigurací se zásadně neliší. Intuitivně by se mohlo zdát, že poskytnutí většího okolí se projeví nalezením kvalitnějšího prahu. Větší okolí se však projeví také větším počtem vstupů CGP. To zapříčíní zvětšení prohledávaného prostoru. Proto by k nalezení lepších řešení byl zřejmě třeba větší počet generací evolučního algoritmu.



Obrázek 6.10: Výsledná F_{score} nejlepších jedinců nalezených s poskytnutím různě velkého okolí pixelu.

6.2.4 Zkoumání vlivu aproximace

V této podsekci jsou rozebrány výsledky experimentů testujících vliv aproximace na parametry nalezených řešení. Je porovnáván vliv na F_{score} a na celkovou plochu obvodu. To vše pro různý tvar mřížky CGP.

Mřížka 1x10

Výsledky experimentů různých kombinací poskytnutých sčítaček a násobiček lze pozorovat na obrázcích 6.11 a 6.12. Obrázek 6.11 poskytuje rozdíly v kvalitě prahování nalezených nejlepších řešení. Původním předpokladem bylo, že se po poskytnutí součástek s vyšším stupněm aproximace bude kvalita nalezených řešení vzhledem k F_{score} snižovat. Výsledky experimentů však tuto hypotézu nepotvrzují.

Jak je vidět u většiny poskytnutých násobiček, kvalita nalezených řešení s vyšším stupněm aproximace sčítaček zpočátku skutečně klesá. U sčítačky s hodnotu $MRE = 1.8\,\%$ se však kvalita nalezeného řešení začíná zlepšovat. Kvalita nalezených řešení po poskytnutí sčítaček s hodnotou $MRE \in \{2.93\,\%, 6.23\,\%, 14.58\,\%, \}$ bývá dokonce lepší, než u sčítačky přesné. Po dosáhnutí příliš vysokého stupně aproximace (MRE sčítačky větší než $14.58\,\%$) kvalita nalezených řešení opět prudce klesne.

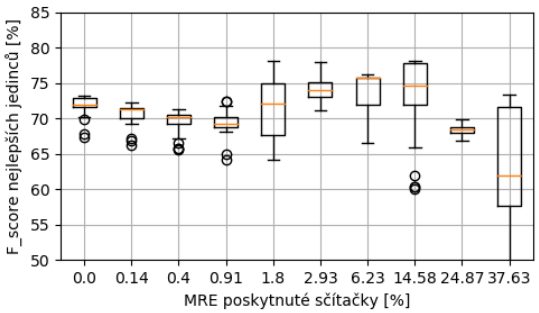
Výjimkou mezi násobičkami je násobička mul8u_13QR. Zanedbáme-li případy sčítaček s vyšší hodnotu MRE než $14.58\,\%$, jsou pro zbylé sčítačky výsledky srovnatelné. Oproti jiným násobičkám jsou výsledky přesných sčítaček dokonce lepší. Přestože je u této násobičky hodnota $MRE = 44\,\%$, kvalita nalezených jedinců překonává přesnou násobičku mul8u_1JFF.

Rozdíly v celkové ploše obvodů pro nalezená nejlepší řešení lze pozorovat na obrázku 6.12. Předpokládaným trendem bylo snižování celkové plochy obvodu s poskytnutím součástky s vyšším stupněm aproximace. Tento trend lze pozorovat také na výsledcích experimentů. Plocha skutečně postupně klesá. Po dosažení sčítačky add8u_08V však plocha rapidně vzroste. To lze vysvětlit tím, že je zde příliš nepřesná sčítačka kompenzována násobičkou, jejíž plocha je řádově větší. U násobiček, které jsou aproximovány více než mul8_u2AC, se již tento jev neobjevuje. Lze to odůvodnit tím, že poskytnutá sčítačka i násobička jsou již příliš nepřesné a použity minimálně, případně vůbec.

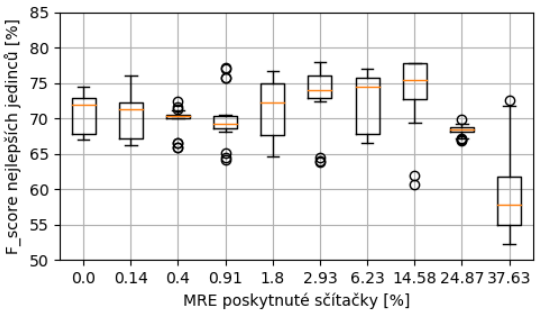
Porovnání výsledků pro jednotlivé kombinace sčítaček a násobiček vzhledem k F_{score} i celkové ploše lze pozorovat na obrázku 6.13. Pro přehlednost zde nejsou zobrazeny výsledky pro násobičky mul8u_EXZ a mul8u_150Q, jejichž průběhy jsou z předchozího zkoumání (obrázky 6.11 a 6.12) srovnatelné s přesnou násobičkou mul8u_1JFF. Z jednoho běhu evolučního algoritmu je výstupem nejlepší nalezený jedinec vzhledem k F_{score}. Ze všech nezávislých běhů dané konfigurace je z těchto jedinců zobrazena Pareto–fronta vzhledem k F_{score} a celkové ploše obvodu. Pro přehlednost je na obrázku 6.13 zobrazena obrácená hodnota F_{score} – tedy $1/F_{score}$. Díky tomu lze za lepší řešení považovat to, jehož hodnota je nižší na obou osách.

Opět lze pozorovat podobný trend jako v předchozích případech. Řešení nalezená s poskytnutím aproximovaných sčítaček bývají lepší, než řešení s poskytnutím sčítačky přesné (tmavě modrá). Zhoršení přichází až u sčítaček s hodnotou $MRE = 24.87\,\%$ (hnědá) a $MRE = 37.63\,\%$ (růžová). Sčítačky s hodnotami $MRE \in \{1.8\,\%, 2.93\,\%, 6.23\,\%, 14.58\,\%\}$ nalézají řešení kvalitnější.

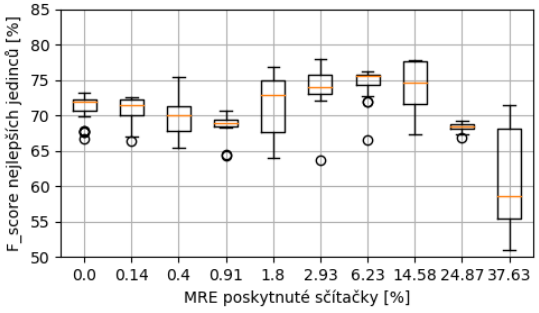
S rostoucí hodnotou MRE poskytnutých násobiček lze pozorovat snižování celkové plochy obvodu při zachování stejné kvality prahování a nalezení více nedominovaných řešení.



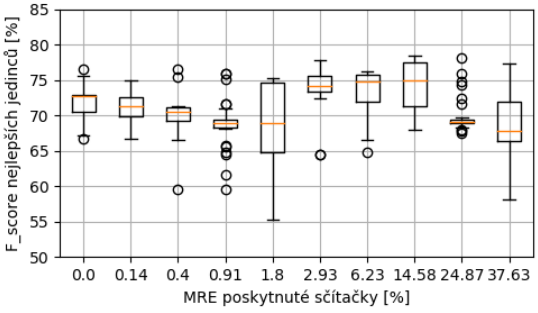
Násobička *mul8u_1JFF*



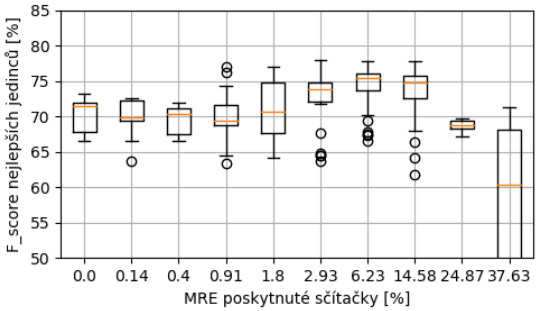
Násobička *mul8u_EXZ*



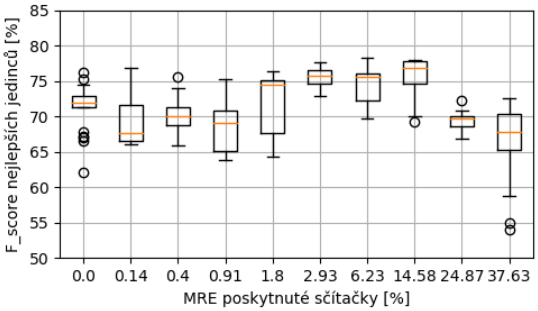
Násobička *mul8u_150Q*



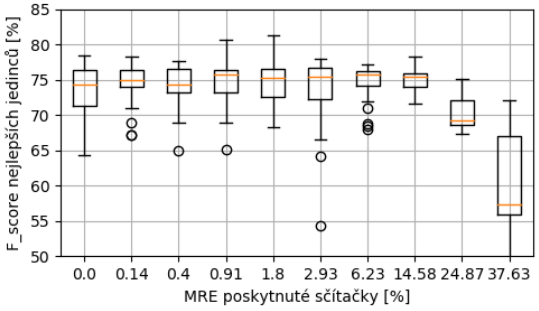
Násobička *mul8u_2AC*



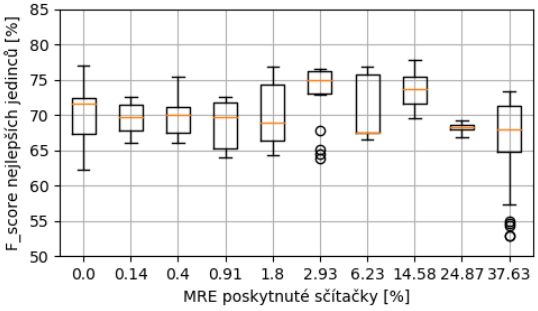
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

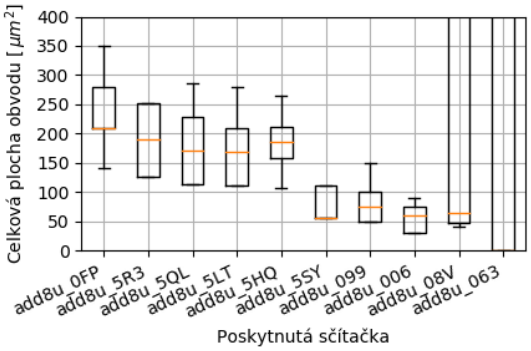


Násobička *mul8u_13QR*

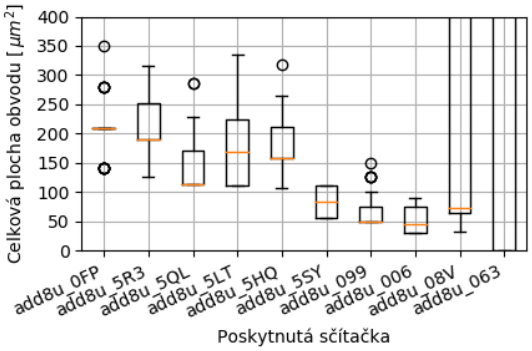


Násobička *mul8u_199Z*

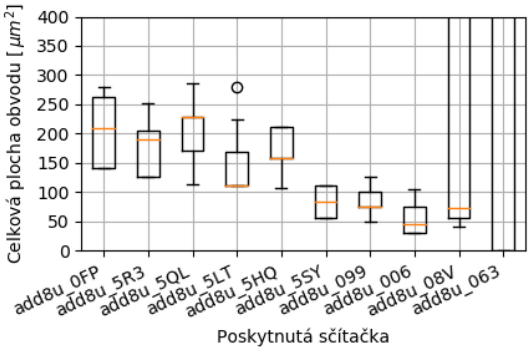
Obrázek 6.11: Výsledky nejlepších jedinců vzhledem k F_{score}. Uvedená F_{score} je počítána pro celý dataset. Případy, kdy se F_{score} pro dataset lišila od trénovací o více než 20 %, nejsou zahrnuty. Tato řešení byla nalezena s tvarem mřížky CGP 1x10.



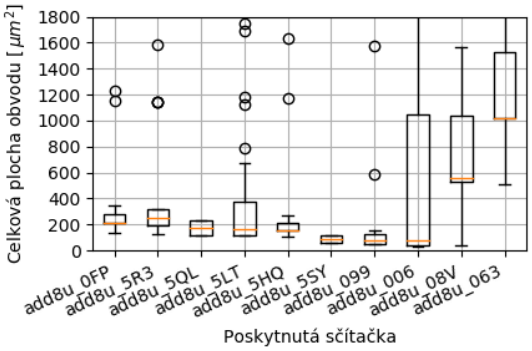
Násobička *mul8u_1JFF*



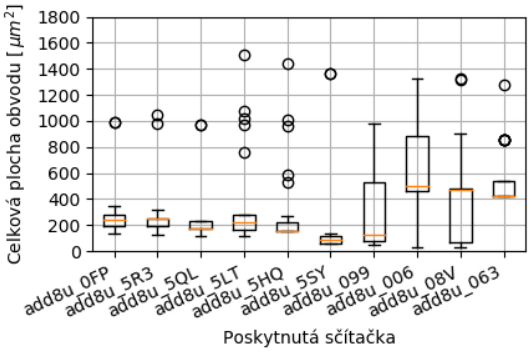
Násobička *mul8u_EXZ*



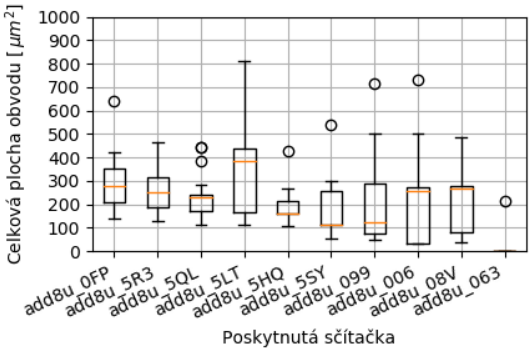
Násobička *mul8u_150Q*



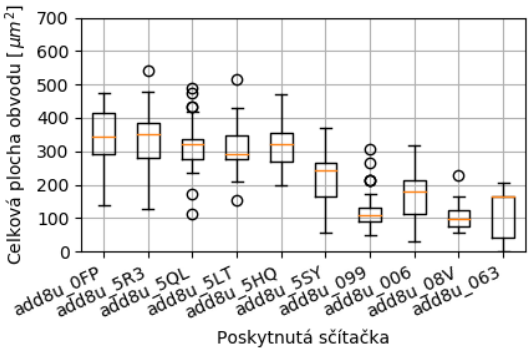
Násobička *mul8u_2AC*



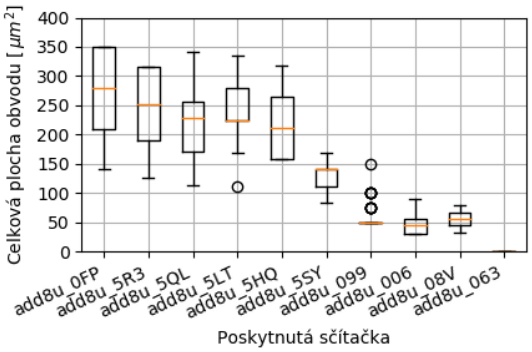
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

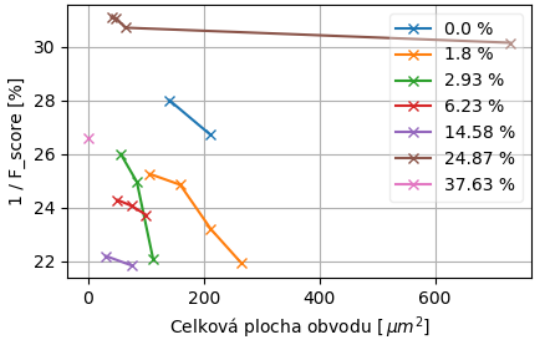


Násobička *mul8u_13QR*

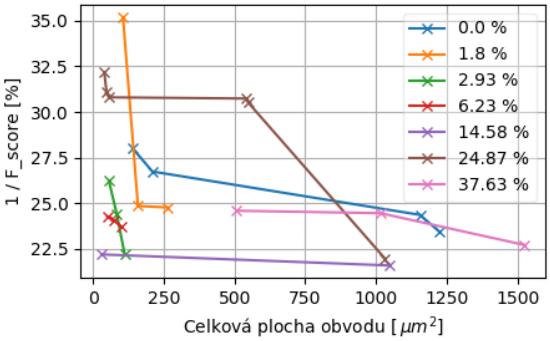


Násobička *mul8u_199Z*

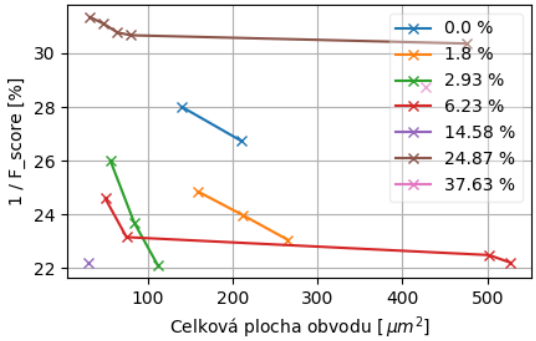
Obrázek 6.12: Celková plocha jedinců s nejvyšší kvalitou prahování. Zobrazeny výsledky pro jednotlivé kombinace sčítaček a násobiček. Uvedení jedinci byli nalezeni s tvarem mřížky CGP 1x10.



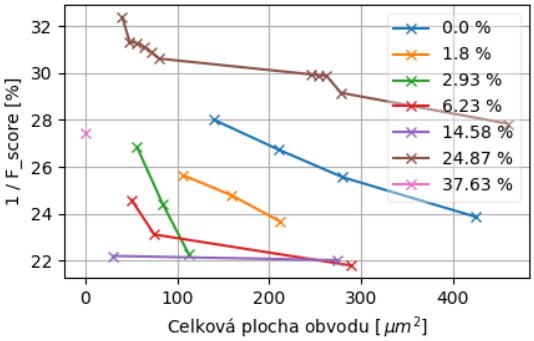
Násobička *mul8u_1JFF*



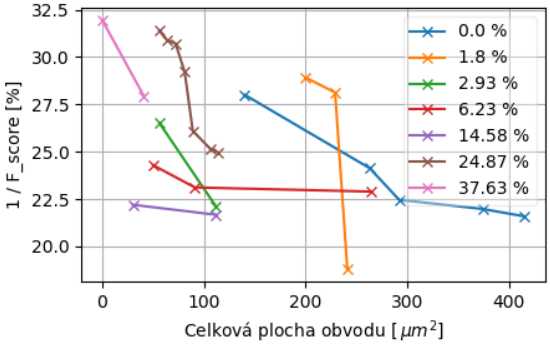
Násobička *mul8u_2AC*



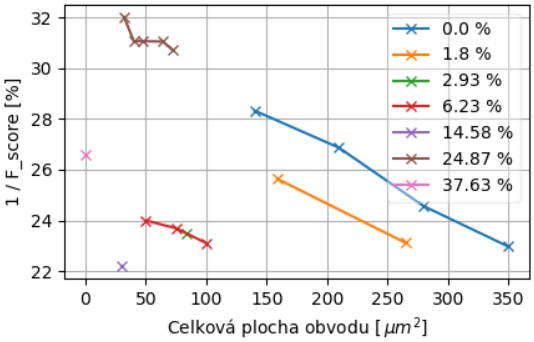
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*



Násobička *mul8u_13QR*

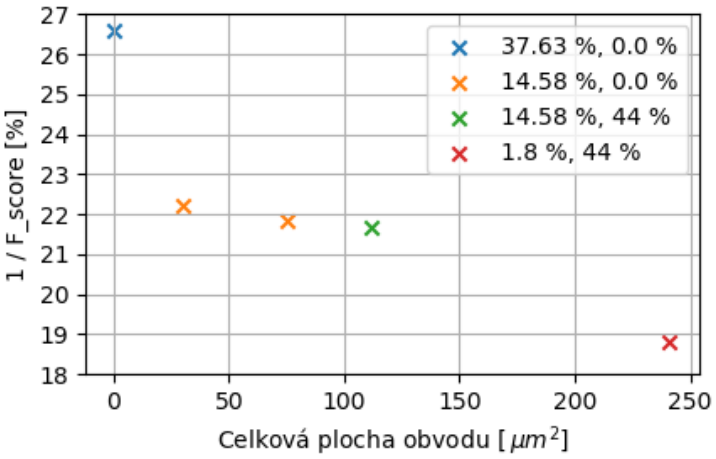


Násobička *mul8u_199Z*

Obrázek 6.13: Znázornění řešení, která v rámci jedné konfigurace experimentu nejsou dominována vzhledem k $1/F_{score}$ a celkové ploše obvodu. Lepší řešení tedy představuje řešení s nižší hodnotou na obou osách. Uvedeny výsledky pro vybrané kombinace sčítaček a násobiček. Legenda udává hodnotu MRE užité sčítačky. Tato řešení byla nalezena pro mřížku CGP 1x10.

Na obrázku 6.14 je zobrazena Pareto–fronta vzhledem k $1/F_{score}$ a celkové ploše nalezených obvodů. Jde o nejlepší řešení napříč všemi kombinacemi poskytnutých sčítaček a násobiček. Mezi těmito řešeními se vůbec nevyskytuje případ poskytnutí přesné sčítačky a přesné násobičky. Je vidět, že nejlepších řešení je dosaženo s poskytnutím alespoň jedné nepřesné součástky. Zajímavé je, že násobičky se zde objevují pouze dvě. Násobička přesná (mul8u_1JFF) a násobička s hodnotou $MRE = 44\,\%$ (mul8u_13QR). Co se týče sčítaček,

je zde nejvíce zastoupena sčítačka s $MRE = 14.58\,\%$. Nejkvalitnější řešení z pohledu F_{score} je však nalezeno s pomocí sčítačky s $MRE = 1.8\,\%$.



Obrázek 6.14: Pareto–fronta nejlepších řešení napříč různými kombinacemi poskytnuté sčítačky a násobičky. Legenda udává hodnoty MRE poskytnutých součástek. První v pořadí uvedena sčítačka, poté násobička. Řešení nalezena s mřížkou CGP 1x10.

Mřížka 5x10

Výsledky experimentů různých kombinací poskytnutých sčítaček a násobiček pro mřížku CGP ve tvaru 5x10 lze pozorovat na obrázcích 6.15, 6.16 a 6.17.

Obrázek 6.15 poskytuje rozdíly v kvalitě prahování nalezených nejlepších řešení. Stejně jako u případu mřížky 1x10 se nepotvrzuje snižování kvality s vyšším stupněm aproximace. Zpočátku sice kvalita s nepřesnějšími sčítačkami klesá, již od hodnoty $MRE = 1.8\,\%$ však opět vzroste nad kvalitu sčítačky přesné. Při dosažení příliš vysoké aproximace ($MRE > 14.58\,\%$) kvalita opět klesne.

Tento trend lze pozorovat u všech poskytnutých násobiček. Tomuto se opět vymiká násobička mul8u_13QR. V kombinaci s touto násobičkou je kvalita řešení pro jednotlivé sčítačky s hodnotou $MRE < 24.87\,\%$ srovnatelná. Sčítačky nejpřesnější ($MRE < 1\,\%$) zde opět vykazují kvalitnější výsledky, než v kombinaci s jinými násobičkami. Medián F_{score} nalezených řešení dosahuje u těchto kombinací kvality prahování $F_{score} = 75\,\%$.

Rozdíly v celkové ploše obvodů pro nalezená nejlepší řešení lze pozorovat na obrázku 6.16. Předpoklad menší plochy obvodu se vzrůstající aproximací se potvrzuje pouze s poskytnutím relativně přesných násobiček. U násobiček mul8u_2AC, mul8u_185Q a mul8u_FTA se však trend obrací. U těchto násobiček se plocha nalezeného obvodu zvětšuje s růstem aproximace sčítaček. Kvalitnější prahování je v těchto případech dosáhnuto na úkor větší plochy obvodu.

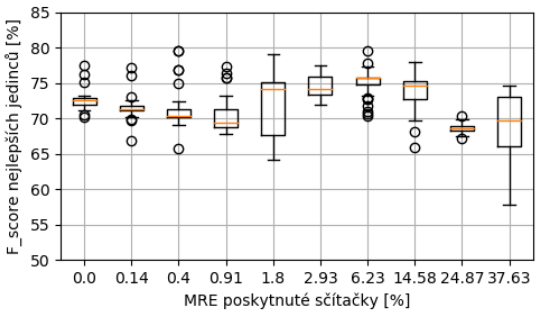
Zajímavé je, že násobička mul8u_13QR opět vykazuje snižování plochy s vyšší aproximací poskytnutých sčítaček. To přesto, že výsledky kombinací s touto násobičkou dosahují nejvyšší F_{score} napříč všemi násobičkami.

Porovnání výsledků pro jednotlivé kombinace sčítaček a násobiček vzhledem k F_{score} i celkové ploše lze pozorovat na obrázku 6.17. Pro přehlednost zde opět nejsou zobrazeny

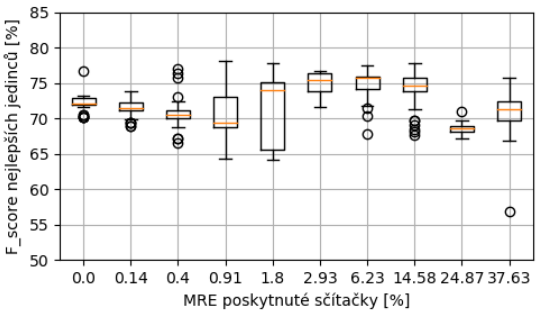
výsledky pro násobičky mul8u_EXZ a mul8u_150Q, jejichž průběhy jsou z předchozího zkoumání (obrázky 6.15 a 6.16) srovnatelné s přesnou násobičkou mul8u_1JFF.

I v případě mřížky 5x10 lze pozorovat nejhorší výsledky nalezených řešení u sčítaček, jejichž hodnota MRE je příliš vysoká (hnědá a růžová). Přesná sčítačka (tmavě modrá) si však v tomto případě vede výrazně lépe. V kombinaci s násobičkami mul8u_FTA a mul8u_13QR dokonce nalézá nejkvalitnější řešení vzhledem k F_{score}.

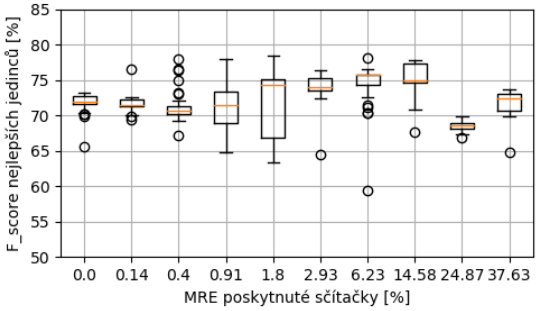
Kombinace přesné násobičky (mul8u_1JFF) s přesnou sčítačkou (tmavě modrá) však opět vykazuje výsledky horší, než kombinace využívající součástky s přiměřenou mírou aproximace.



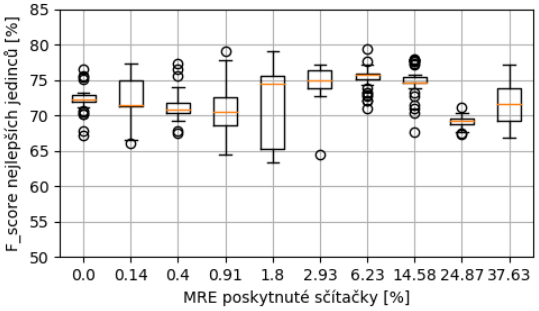
Násobička *mul8u_1JFF*



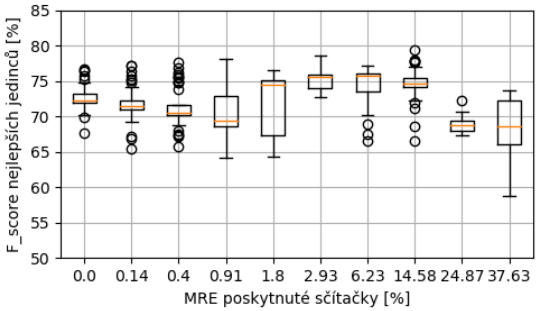
Násobička *mul8u_EXZ*



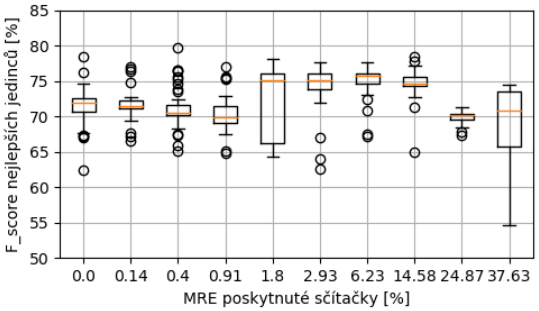
Násobička *mul8u_150Q*



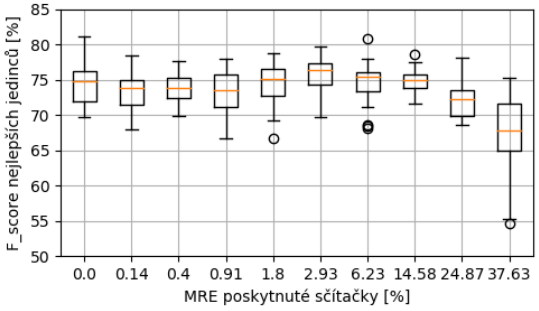
Násobička *mul8u_2AC*



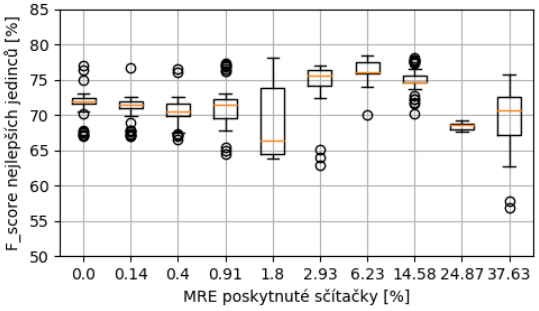
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

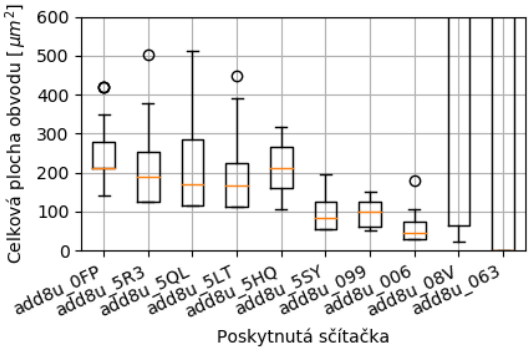


Násobička *mul8u_13QR*

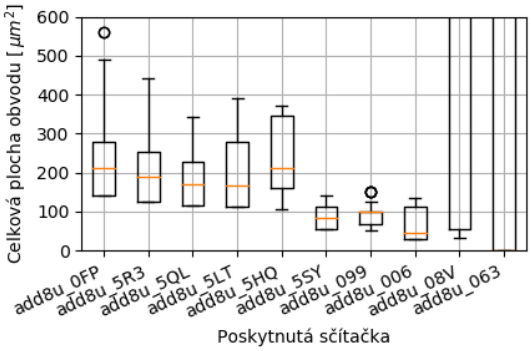


Násobička *mul8u_199Z*

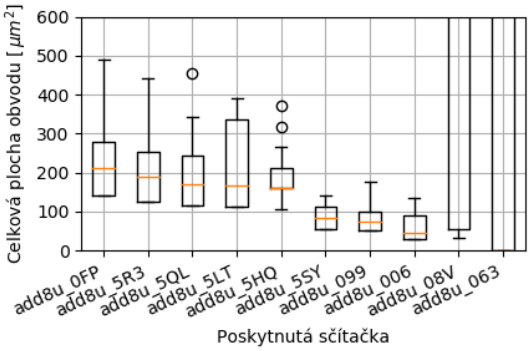
Obrázek 6.15: Výsledky nejlepších jedinců vzhledem k F_{score}. Uvedená F_{score} je počítána pro celý dataset. Případy, kdy se F_{score} pro dataset lišila od trénovací o více než 20 %, nejsou zahrnuty. Tato řešení byla nalezena s tvarem mřížky CGP 5x10.



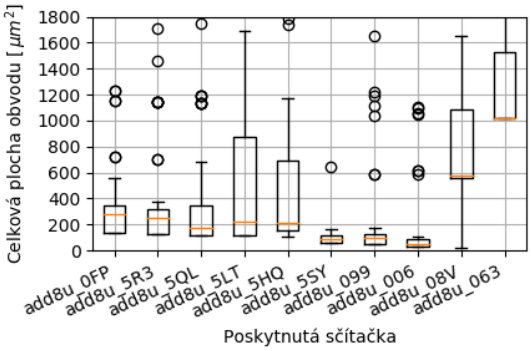
Násobička *mul8u_1JFF*



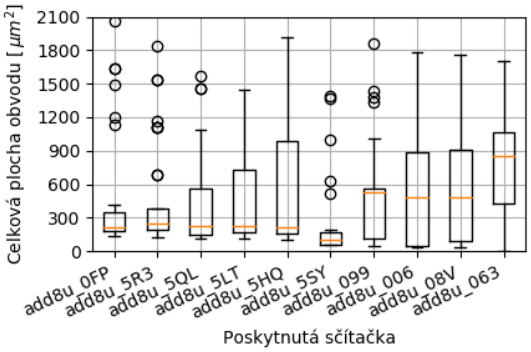
Násobička *mul8u_EXZ*



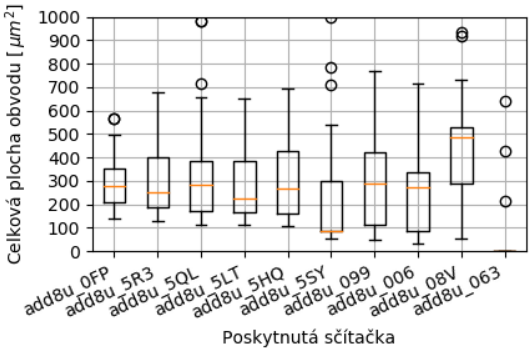
Násobička *mul8u_150Q*



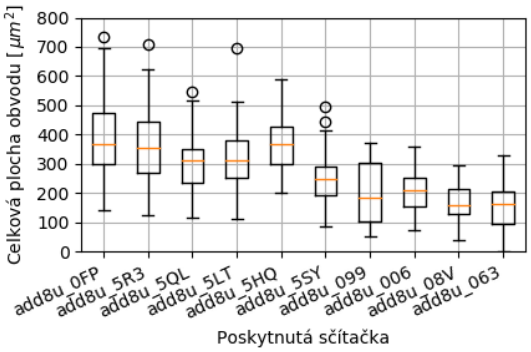
Násobička *mul8u_2AC*



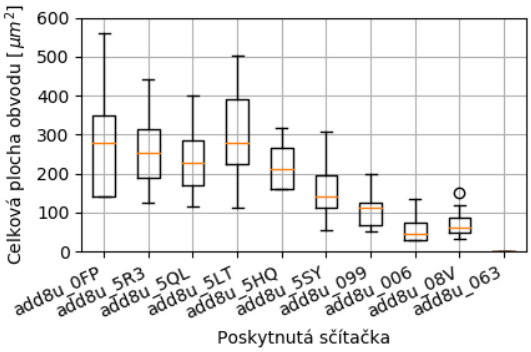
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

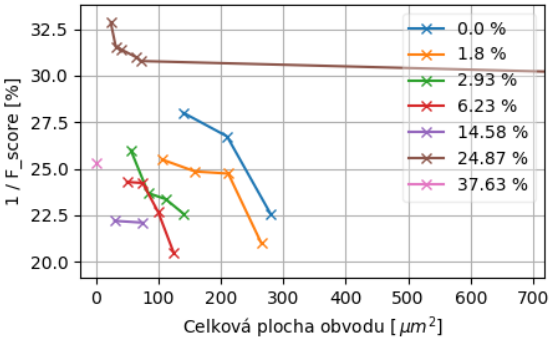


Násobička *mul8u_13QR*

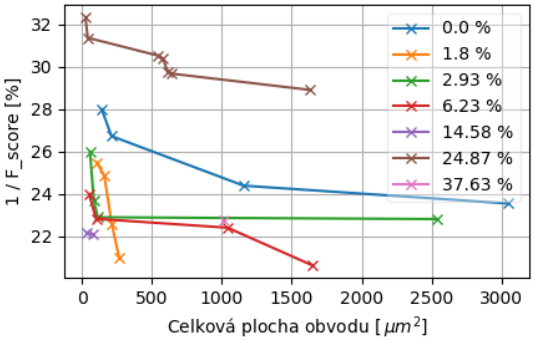


Násobička *mul8u_199Z*

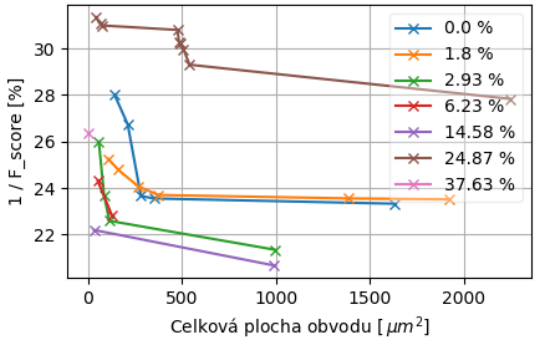
Obrázek 6.16: Celková plocha jedinců s nejvyšší kvalitou prahování. Zobrazeny výsledky pro jednotlivé kombinace sčítaček a násobiček. Uvedení jedinci byli nalezeni s tvarem mřížky CGP 5x10.



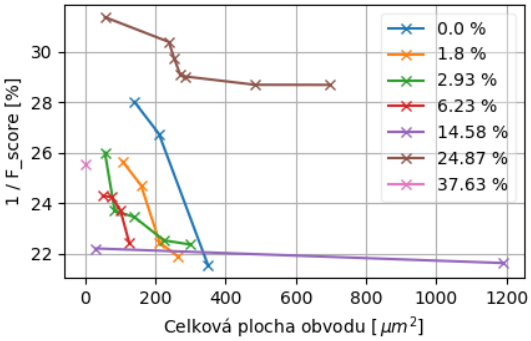
Násobička *mul8u_1JFF*



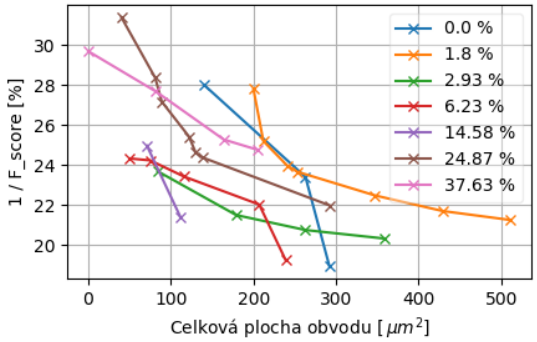
Násobička *mul8u_2AC*



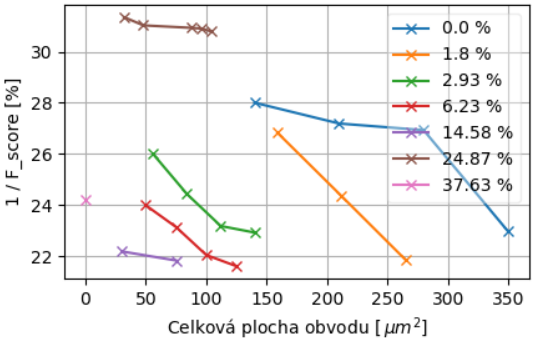
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*



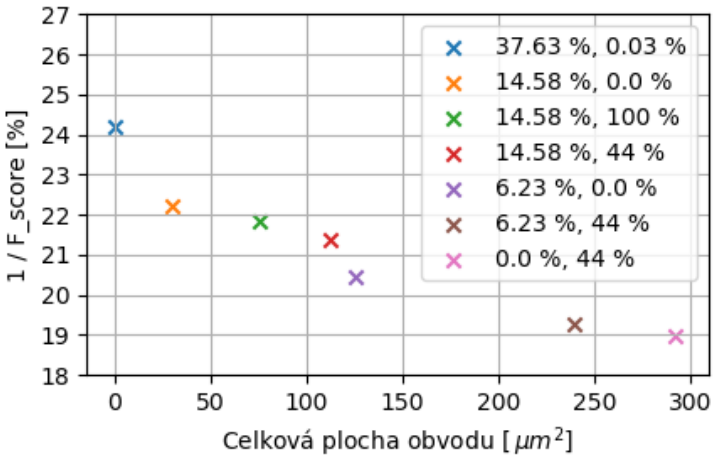
Násobička *mul8u_13QR*



Násobička *mul8u_199Z*

Obrázek 6.17: Znázornění řešení, která v rámci jedné konfigurace experimentu nejsou dominována vzhledem k $1/F_{score}$ a celkové ploše obvodu. Lepší řešení tedy představuje řešení s nižší hodnotou na obou osách. Uvedeny výsledky pro vybrané kombinace sčítaček a násobiček. Legenda udává hodnotu MRE užité sčítačky. Tato řešení byla nalezena pro mřížku CGP 5x10.

Na obrázku 6.18 je zobrazena Pareto–fronta vzhledem k $1/F_{score}$ a celkové ploše nalezených obvodů. Jde o nejlepší řešení napříč všemi kombinacemi poskytnutých sčítaček a násobiček. Opět zde nemá zastoupení kombinace přesných součástek. Nejčastěji jsou z násobiček zastoupeny násobičky s $MRE = 0\,\%$ a $MRE = 44\,\%$. Ze sčítaček jsou to pak sčítačky s hodnotami $MRE \in \{6.23\,\%, 14.58\,\%\}$.



Obrázek 6.18: Pareto–fronta nejlepších řešení napříč různými kombinacemi poskytnuté sčítačky a násobičky. Legenda udává hodnoty MRE poskytnutých součástek. První v pořadí uvedena sčítačka, poté násobička. Řešení nalezena s mřížkou CGP 5x10.

Mřížka 10x10

Výsledky experimentů pro tvar mřížky CGP 10x10 lze pozorovat na obrázcích 6.19, 6.20 a 6.21.

Zaměříme-li se pouze na kvalitu prahování F_{score} (obrázek 6.19), je opět zřejmý podobný průběh výsledků jako u předchozích tvarů mřížek. Nejlepší výsledky vykazují kombinace se sčítačkami jejichž $MRE \in \{1.8\,\%, 2.93\,\%, 6.23\,\%, 14.58\,\%\}$.

Se zvyšujícím se stupněm aproximace násobiček se výsledky zásadně neliší. Jedinou vymikající se násobičkou je opět mul8u_13QR. Tato násobička vykazuje zlepšení v kombinaci se sčítačkami s nižším MRE. Ostatní násobičky mají co se týče F_{score} srovnatelné výsledky.

Zaměříme-li se na plochu navržených obvodů (obrázek 6.20), mají násobičky mul8u_1JFF, mul8u_EXZ a mul8u_150Q podobný průběh. Jedná se o násobičky relativně přesné ($MRE < 1\,\%$). Zde lze opět pozorovat snižování plochy se vzrůstající aproximací jednotlivých sčítaček. U sčítaček s hodnotou $MRE > 15.58\,\%$ pak plocha radikálně vzroste.

Násobičky mul8u_2AC, mul8u_185Q a mul8u_FTA se však v tomto ohledu liší. Plocha nalezených obvodů v kombinaci s různými sčítačkami je značně nedeterministická. Pro porovnání vhodnosti využití těchto násobiček je třeba se podívat na obě kritéria v rámci nalezeného řešení. Jak na plochu obvodu, tak i F_{score}.

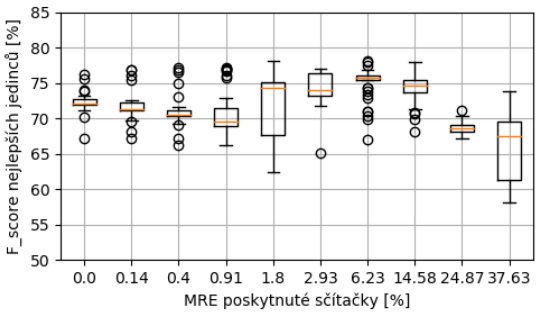
Násobičky mul8u_13QR a mul8u_199Z vykazují co se týče plochy průběhy podobné jako relativně přesné násobičky ($MRE < 1\,\%$). Rozdílem je, že dvě z nejméně přesných sčítaček nalézají v kombinaci s těmito násobičkami plochu nejmenší.

Porovnání výsledků pro jednotlivé kombinace sčítaček a násobiček vzhledem k F_{score} i celkové ploše lze pozorovat na obrázku 6.21. Výsledky násobiček mul8u_EXZ a mul8u_150Q, jejichž průběhy jsou z předchozího zkoumání (obrázky 6.19 a 6.20) srovnatelné s přesnou násobičkou mul8u_1JFF jsou zanedbány. Podobně nejsou vykresleny výsledky kombinací se sčítačkami jejichž $MRE < 1\,\%$, jelikož jsou téměř totožné se sčítačkou, jejíž $MRE = 0\,\%$ (tato je zobrazena modře).

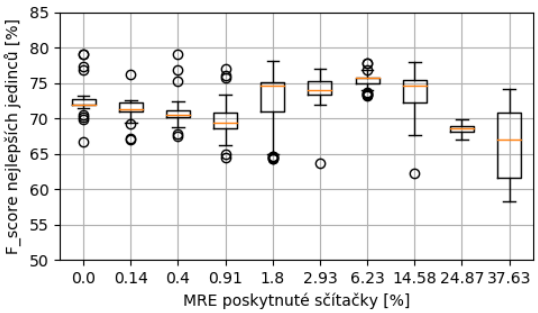
Pomineme-li případy sčítaček s příliš vysokým stupněm aproximace ($MRE > 14.58\,\%$), jsou výsledky ostatních kombinací značně vyrovnanější, než je tomu u řešení s mřížkou CGP 1x10 a 5x10. S narůstající aproximací násobiček se kvalita nalezených řešení vyrovnává.

Porovnáme-li výsledky různých sčítaček, lze opět pozorovat, že sčítačky s rozumnou mírou aproximace ($MRE \in \{l.8\,\%, 2.93\,\%, 6.23\,\%, 14.58\,\%\}$) vykazují kvalitnější výsledky, než sčítačka přesná (tmavě modrá).

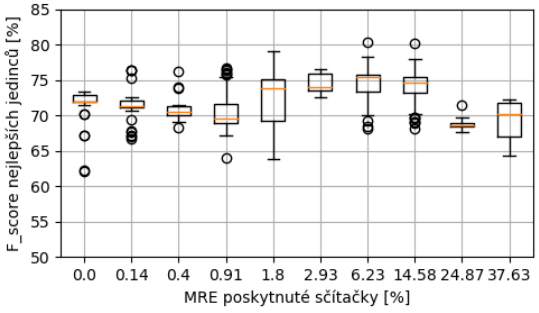
Nejlepší nalezená řešení jsou v případě mřížky 10x10 kvalitnější, než u předchozích případů 5x10 a 1x10.



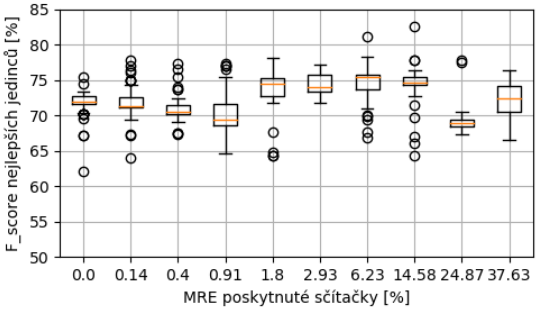
Násobička *mul8u_1JFF*



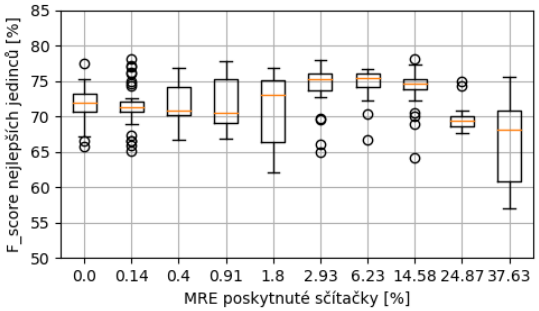
Násobička *mul8u_EXZ*



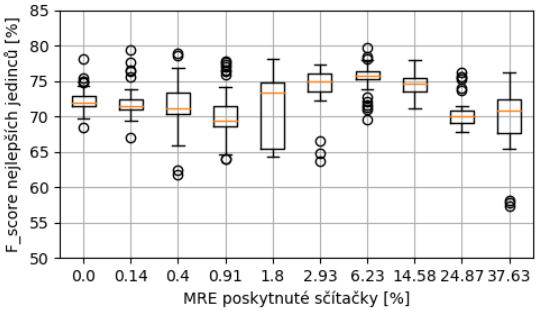
Násobička *mul8u_150Q*



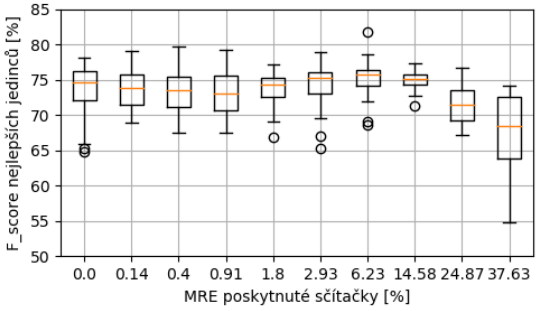
Násobička *mul8u_2AC*



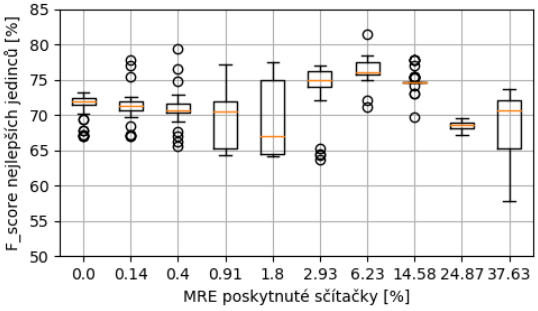
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

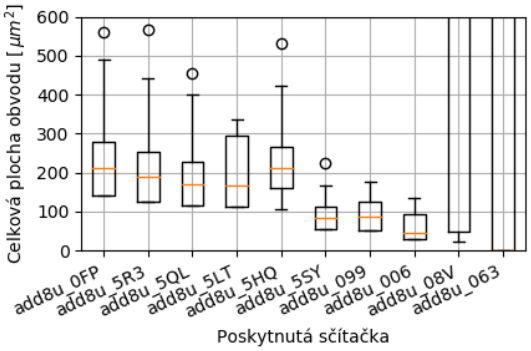


Násobička *mul8u_13QR*

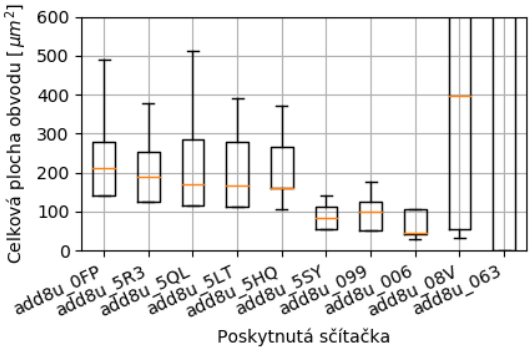


Násobička *mul8u_199Z*

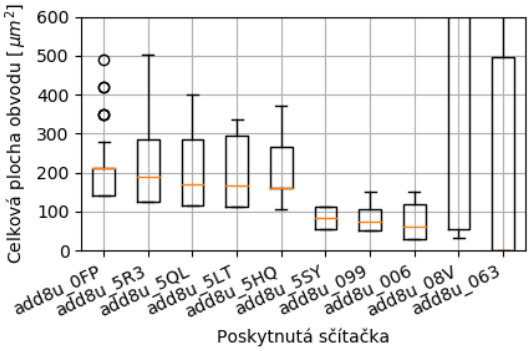
Obrázek 6.19: Výsledky nejlepších jedinců vzhledem k F_{score}. Uvedená F_{score} je počítána pro celý dataset. Případy, kdy se F_{score} pro dataset lišila od trénovací o více než 20 %, nejsou zahrnuty. Tato řešení byla nalezena s tvarem mřížky CGP 10x10.



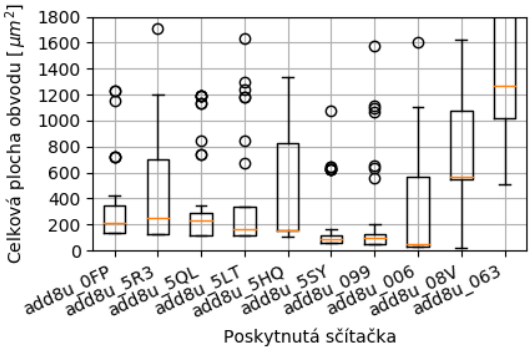
Násobička *mul8u_1JFF*



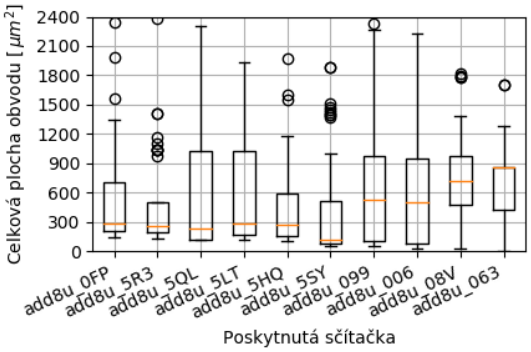
Násobička *mul8u_EXZ*



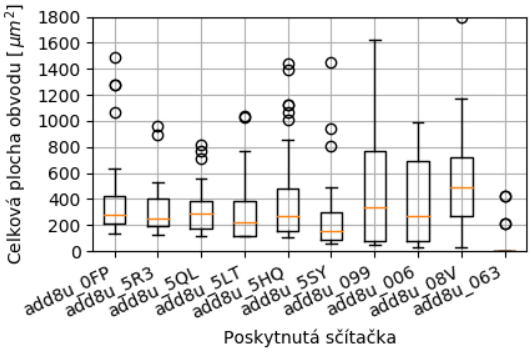
Násobička *mul8u_150Q*



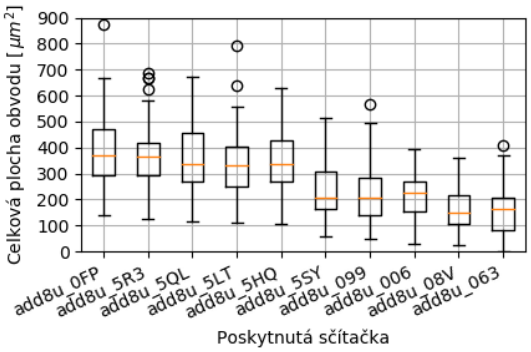
Násobička *mul8u_2AC*



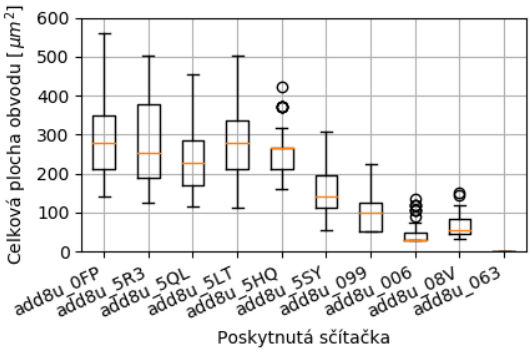
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*

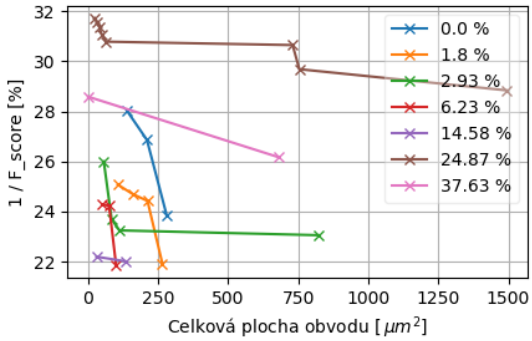


Násobička *mul8u_13QR*

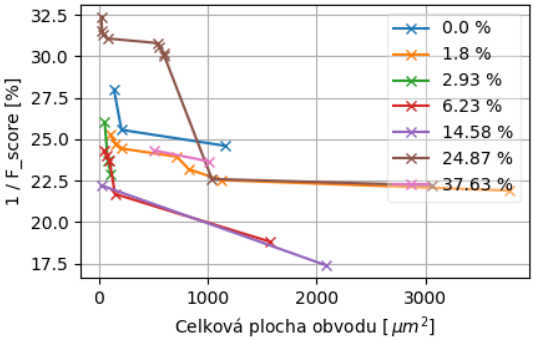


Násobička *mul8u_199Z*

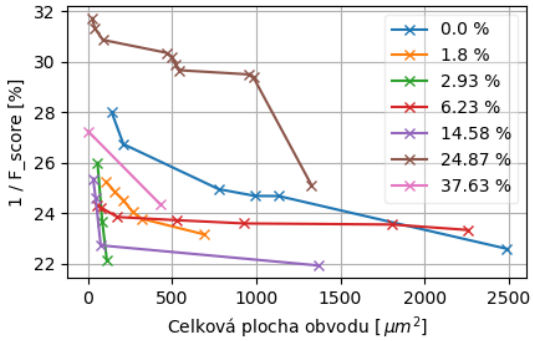
Obrázek 6.20: Celková plocha jedinců s nejvyšší kvalitou prahování. Zobrazeny výsledky pro jednotlivé kombinace sčítaček a násobiček. Uvedení jedinci byli nalezeni s tvarem mřížky CGP 10x10.



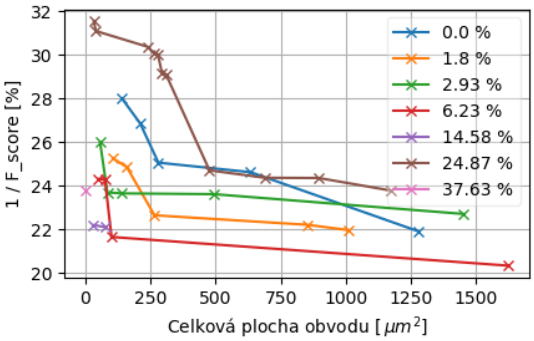
Násobička *mul8u_1JFF*



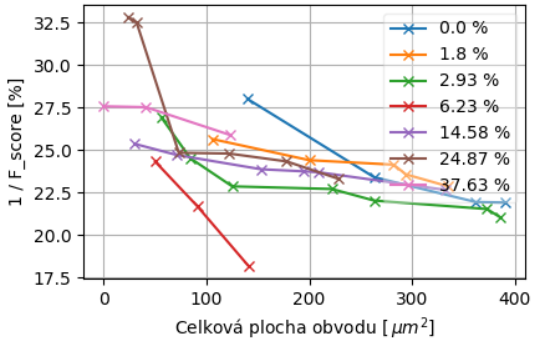
Násobička *mul8u_2AC*



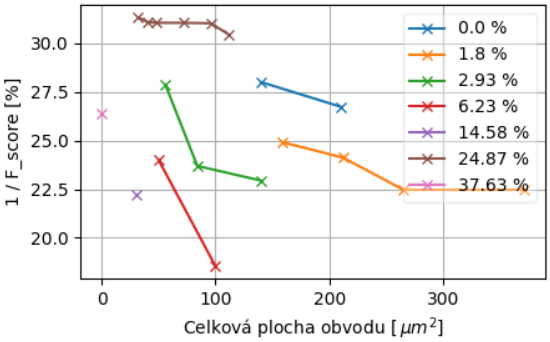
Násobička *mul8u_185Q*



Násobička *mul8u_FTA*



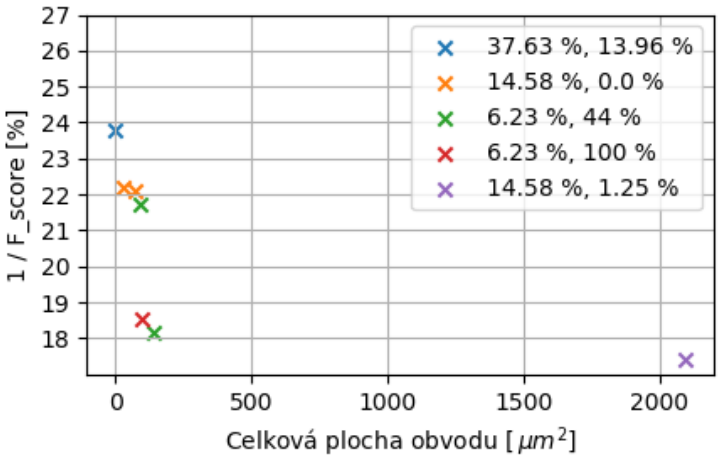
Násobička *mul8u_13QR*



Násobička *mul8u_199Z*

Obrázek 6.21: Znázornění řešení, která v rámci jedné konfigurace experimentu nejsou dominována vzhledem k $1/F_{score}$ a celkové ploše obvodu. Lepší řešení tedy představuje řešení s nižší hodnotou na obou osách. Uvedeny výsledky pro vybrané kombinace sčítaček a násobiček. Legenda udává hodnotu MRE poskytnuté sčítačky. Tato řešení byla nalezena pro mřížku CGP 10x10.

Na obrázku 6.22 lze pozorovat nejlepší řešení napříč všemi kombinacemi poskytnutých součástek. V tomto případě se zde objevuje více případů násobiček. Opět však nechybí násobičky s $MRE \in \{0\,\%, 44\,\%\}$. Ze sčítaček zde opět mají nejčastější zastoupení sčítačky s $MRE \in \{6.23\,\%, 14.58\,\%\}$. Tyto dvě sčítačky dosahují nejlepších výsledků napříč různými tvary mřížky CGP.



Obrázek 6.22: Pareto–fronta nejlepších řešení napříč různými kombinacemi poskytnuté sčítačky a násobičky. Legenda udává hodnoty MRE poskytnutých součástek. První v pořadí uvedena sčítačka, poté násobička. Řešení nalezena s mřížkou CGP 10x10.

6.2.5 Výstupy nejlepších jedinců

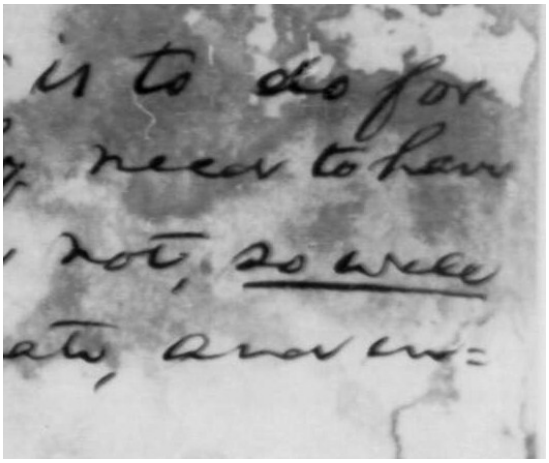
V tabulce 6.2 jsou vypsána nejlepší nalezená řešení napříč všemi experimenty. Jsou vybrány z různých kombinací poskytnutých sčítaček a násobiček a z různých případů mřížky CGP. Je zde také uvedeno nejlepší nalezené řešení po poskytnutí přesných součástek.

Je vidět, že po poskytnutí přesných součástek je kvalita nalezeného řešení z hlediska F_{score} nižší. V některých případech je horší také plocha nalezeného obvodu.

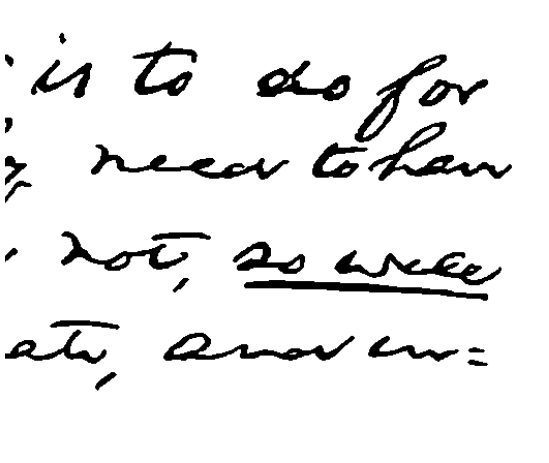
Výstupy některých řešení z tabulky 6.2 lze pozorovat na obrázku 6.23. Lze si povšimnout zhoršení prahování v oblasti světlého „fleku“. Přesné součástky si v tomto místě vedou hůře, než součástky aproximované.

	Sčítačka	**MRE**	**Násobička**	**MRE**	F_{score}	**Plocha**	**Mřížka**
1)	add8u_006	14.58 %	mul8u_2AC	1.25 %	82.61 %	2092 μm^2	10x10
2)	add8u_099	6.23 %	mul8u_13QR	44 %	81.83 %	141 μm^2	10x10
3)	add8u_099	6.23 %	mul8u_199Z	100 %	81.46 %	100 μm^2	10x10
4)	add8u_5HQ	1.8 %	mul8u_13QR	44 %	81.20 %	241 μm^2	5x10
5)	add8u_0FP	0 %	mul8u_13QR	44 %	81.05 %	292 μm^2	10x10
6)	add8u_0FP	0 %	mul8u_1JFF	0 %	77.44 %	280 μm^2	5x10

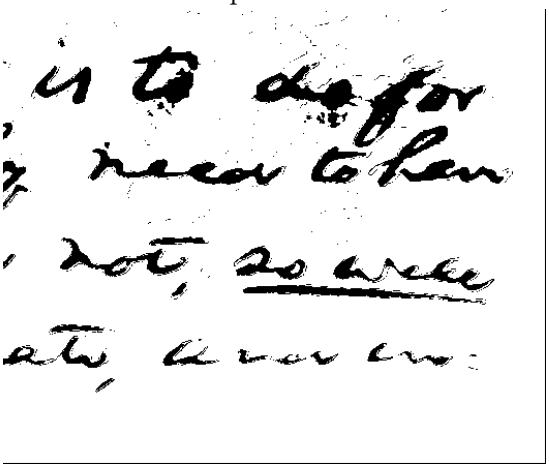
Tabulka 6.2: Tabulka výsledků nejlepších jedinců.



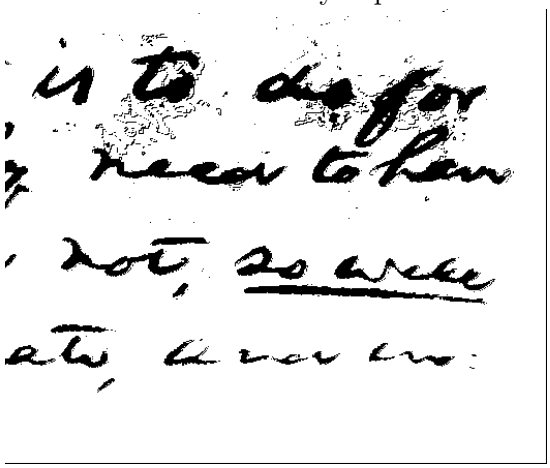
Vstupní obrázek



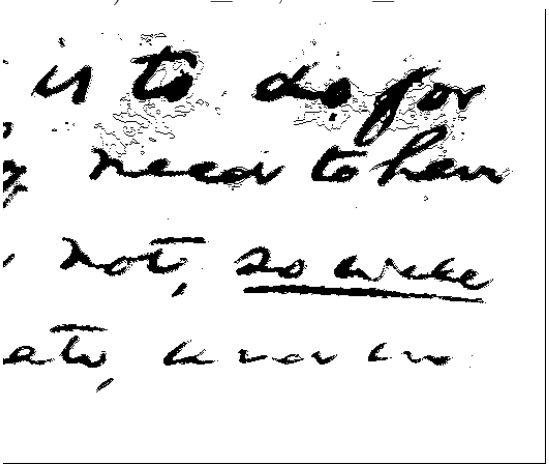
Referenční výstup



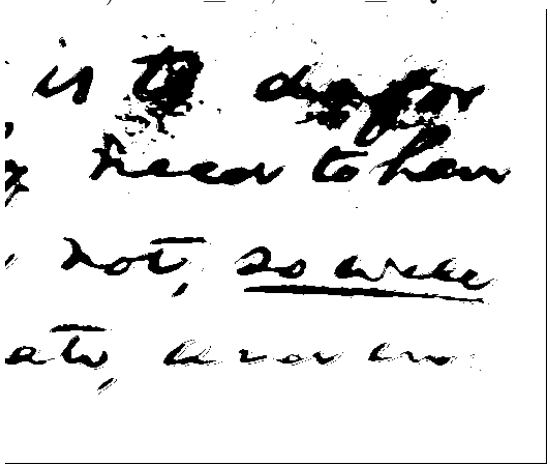
1) add8u_006, mul8u_2AC



2) add8u_099, mul8u_13QR



3) add8u_099, mul8u_199Z



6) add8u_0FP, mul8u_1JFF

Obrázek 6.23: Výstupy prahování pomocí nejlepších jedinců uvedených v tabulce 6.2. Pro zvýraznění rozdílů oříznuta a zvětšena pouze část obrázku. Uvedeny výstupy pro řešení 1), 2), 3) a řešení s přesnými součástkami 6). Černý okraj vyprahovaného obrázku je při výpočtu F_{score} zanedbáván.

Kapitola 7

Závěr

Předmětem této práce bylo aplikování přibližného počítání na oblast zpracování obrazu. Bylo zjišťováno, jaký bude mít aproximace vliv na výslednou kvalitu výstupu. Jestli se kvalita zhorší, což bylo předpokládáno, nebo se radikálně nezhorší a bude vykazovat zlepšení v jiném ohledu.

Z oblasti zpracování obrazu bylo vybráno téma adaptivního prahování psaného textu. Byla získána sada referenčních vstupů a výstupů.

Pro automatizovaný přístup k aproximaci byly využity evoluční algoritmy, konkrétně CGP+ES a NSGA-II.

Byly provedeny experimenty s dvěma odlišnými přístupy – aproximace existujícího algoritmu a návrh algoritmu s poskytnutím aproximovaných součástek.

Výsledky prvního přístupu vykazují očekávané zhoršení výsledku prahování po zvýšení úrovně aproximace algoritmu. Pro některé nízké úrovně aproximace je však kvalita výsledku srovnatelná. Zdá se tedy, že v tomto případě využití aproximací vhodné není.

Výsledky druhého přístupu, tedy návrh nového algoritmu (obvodu) po poskytnutí již aproximovaných součástek však vykazují výsledky rozdílné. Předpoklad o zhoršování kvality výstupu se zvyšujícím se stupněm aproximace je vyvrácen. Samotná kvalita výstupu se po zvyšování aproximace poskytnutých součástek zlepšuje. To až k dosažení hraničního stupně aproximace, kdy kvalita výstupu prudce klesne.

Aproximovaná řešení druhého přístupu nevykazují zlepšení jen co se týče kvality výstupu, ale také u parametru celkové plochy navrženého obvodu.

Součástky jež byly evoluci poskytnuty byly převzaty z *EvoApproxLib*. Byly vybrány aproximované sčítačky a násobičky reprezentující jednotlivé stupně aproximace. Nejlepší výsledky vykazují sčítačky add8u_5HQ, add8u_5SY, add8u_099 a add8u_006 jejichž hodnota MRE je v rozsahu $\langle 1.80\,\%, 14.58\,\% \rangle$. Poskytnuté násobičky vykazují výsledky srovnatelné. Výjimkou je násobička mul8u_13QR ($MRE = 44\,\%$), jež nejčastěji nalézá nejkvalitnější řešení. Poskytnutí násobičky s příliš vysokou aproximací (mul8u_199Z) se však již jeví jako nevyhovující, protože násobička již není ve výsledku využita.

Aplikace přibližného počítání technikou návrhu algoritmu pomocí CGP+ES a poskytnutí aproximovaných součástek se jeví slibně a zřejmě by se tato technika dala aplikovat na více oblastí ve zpracování obrazu.

Co se týče adaptivního prahování psaného textu, je zde prostor ke zlepšení v získání trénovací podmnožiny z celého datasetu. V této práci je tohoto dosahováno čistě náhodným výběrem míst s důrazem na rovnoměrné zastoupení míst reprezentujících popředí a pozadí. Zřejmě by však bylo vhodnější vybírat místa, která jsou nějakým způsobem pro da-

taset specifická. Tím by mohlo být dosaženo trénovací podmnožiny datasetu, která dataset reprezentuje vhodněji a pomocí menšího počtu míst.

Zadání této práce bylo splněno v plném rozsahu.

Literatura

[1] Beyer, H. Evolution strategies. *Scholarpedia*. 2007, roč. 2, č. 8, s. 1965.

[2] Brabazon, A. *Natural computing algorithms*. Heidelberg: Springer, 2015. ISBN 978-3-662-43631-8.

[3] Bradley, D. a Roth, G. Adaptive Thresholding using the Integral Image. *J. Graphics Tools*. 2007, roč. 12, s. 13–21.

[4] Deb, K., Pratap, A., Agarwal, S. a Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. Duben 2002, roč. 6, č. 2, s. 182–197. ISSN 1941-0026.

[5] Eiben, A. E. *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. ISBN 978-3-662-05094-1.

[6] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters*. 2006, roč. 27, č. 8, s. 861 – 874. ROC Analysis in Pattern Recognition. Dostupné z: http://www.sciencedirect.com/science/article/pii/S016786550500303X. ISSN 0167-8655.

[7] Khadra, M. A. B. An introduction to approximate computing. *ArXiv*. 2017.

[8] Kvasnička, V., Pospíchal, J. a Tiňo, P. *Evolučné algoritmy*. Slovenská technická univerzita, 2000. Edícia vysokoškolskỳch učebníc / Slovenská technická univerzita. ISBN 9788022713771.

[9] Mrázek, V., Hrbáček, R., Vašíček, Z. a Sekanina, L. EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods. In: *Proc. of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. European Design and Automation Association, 2017, s. 258–261. Dostupné z: https://www.fit.vut.cz/research/publication/11262. ISBN 978-3-9815370-9-3.

[10] Pratikakis, I., Zagoris, K. a Karagiannis, X. *Document Image Binarization Competition* [online]. Dostupné z: https://vc.ee.duth.gr/dibco2019/.

[11] Sekanina, L., Vasicek, Z. a Mrazek, V. Approximate Circuits in Low-Power Image and Video Processing: The Approximate Median Filter. *Radioengineering*. Září 2017, roč. 26, s. 623–632.

[12] Sekanina, L. Introduction to Approximate Computing: Embedded Tutorial. In: *19th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and*

Systems. Institute of Electrical and Electronics Engineers, 2016, s. 90–95. Dostupné z: https://www.fit.vut.cz/research/publication/11075. ISBN 978-1-5090-2467-4.

[13] Tan, K. C. *Multiobjective evolutionary algorithms and applications*. London: Springer, 2005. ISBN 978-1-84628-132-7.

[14] Vasicek, Z., Bidlo, M. a Sekanina, L. Evolution of efficient real-time non-linear image filters for FPGAs. *Soft Computing*. Listopad 2013, roč. 17.

[15] Vasicek, Z. a Sekanina, L. Evolutionary Approach to Approximate Digital Circuits Design. *Evolutionary Computation, IEEE Transactions on*. Červen 2015, roč. 19, s. 432–444.

Příloha A

Manuál k vytvořeným programům

A.1 Program nsga2

A.1.1 Překlad

Pro překlad je připraven odpovídající `Makefile`. Překlad lze tedy provést příkazem `make`.

A.1.2 Možnosti spuštění

Samotný program lze spustit dvěma způsoby. Prvním z nich je spuštění evoluce, hledající nejlepší chromozom reprezentující aproximovaný algoritmus. Druhým pak samotné prahování obrázku pomocí zadaného chromozomu.

Evoluce

Parametry evoluce jsou definovány v souboru `input/threshold.in`. Každý řádek tohoto souboru definuje určitý parametr evoluce. Vysvětlení tohoto souboru lze nalézt v souboru `input/README`.

Samotnou evoluci pak lze spustit pomocí následujícího příkazu:

- `./nsga2 < input/threshold.in`

Velikost výřezu obrázku z datasetu, který bude poskytnut evoluci je definována v souboru `data_def.h`. Zde lze definovat následující makra:

- `X` – definuje sloupec levého horního rohu výřezu.
- `Y` – definuje řádek levého horního rohu výřezu.
- `COLS` – definuje šířku výřezu.
- `ROWS` – definuje výšku výřezu.

Prahování

Spustit program za účelem prahování obrázku pomocí určitého chromozomu lze tímto způsobem:

- `./nsga2 -f input.pgm`

Parametr `input.pgm` definuje cestu k obrázku, který má být prahován. Je podporován pouze formát šedotónového obrázku `.pgm`.

Po spuštění se uživatele program dotazuje na hodnotu jednotlivých genů chromozomu.

A.1.3 Výstupní soubory

Evoluce

Výstupem evoluce jsou následující soubory s příponou `.out` (písmeno `N` v názvu souboru zastupuje číslo odpovídajícího procesu):

- `all_popN.out` – v tomto souboru je záznam všech jedinců v evoluci a jejich kritéria.
- `best_popN.out` – v tomto souboru jsou uvedeny parametry jedinců, jenž nejsou v celé evoluci dominovány.
- `final_popN.out` – v tomto souboru jsou uvedeny parametry všech jedinců poslední generace.
- `initial_popN.out` – v tomto souboru je zaznamenána počáteční generace evoluce.
- `paramsN.out` – v tomto souboru jsou zaznamenány parametry, se kterými byla evoluce spuštěna.

Prahování

Výstupem prahování je vyprahovaný obrázek `result.pgm`.

A.2 Program cgp

A.2.1 Překlad

Pro překlad je připraven odpovídající `Makefile`. Překlad lze tedy provést příkazem `make`.

A.2.2 Možnosti spuštění

Samotný program lze spustit dvěma způsoby. Prvním z nich je spuštění evoluce, hledající nejlepší chromozom reprezentující aproximovaný algoritmus. Druhým pak samotné prahování obrázku pomocí zadaného chromozomu.

Evoluce

Spuštění za účelem evolučního návrhu obvodu s poskytnutím aproximovaných součástek je provedeno následujícím způsobem:

- `./cgp [options]`

Volitelnými argumenty pak lze definovat nastavení parametrů evoluce. Jsou to tyto:

- `-g N` – nastaví celkový počet generací na `N`.
- `-r N` – nastaví počet řádků mřížky CGP na `N`.
- `-c N` – nastaví počet sloupců mřížky CGP na `N`.

- `-m N` – nastaví maximální počet mutovaných genů v chromozomu na `N`.
- `-l N` – nastaví l_{back} na `N`.
- `-p N` – nastaví velikost populace na `N`.
- `-v N` – nastaví počet míst z datasetu poskytnutých evoluci na `N`.
- `-h` – zapne zaznamenávání všech nově nalezených nejlepších jedinců.

Prahování

Spustit program za účelem prahování obrázku lze následovně:

- `./cgp -f input.pgm -o output.pgm`

Parametr `input.pgm` definuje cestu k obrázku jenž má být prahován. Parametr `output.pgm` definuje jméno vyprahovaného obrázku.

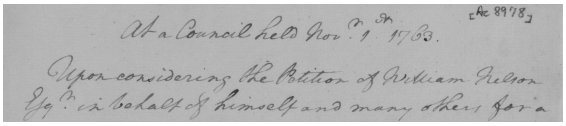
A.2.3 Výstupní soubory

Výstupem evoluce je soubor `log_N.txt`, jenž obsahuje záznam vývoje fitness nejlepšího jedince. Dále chromozom nejlepšího jedince v rámci celé evoluce a jeho F_{score} pro celý dataset. Je zde také uvedena celková plocha nalezeného řešení.

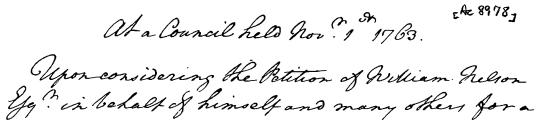
Výstupem prahování je vyprahovaný obrázek s definovaným jménem.

Příloha B

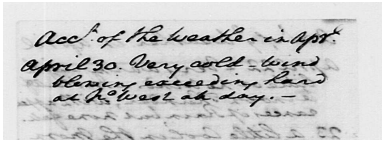
Dataset psaného textu



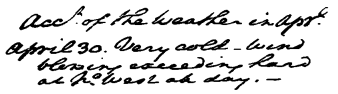
H01



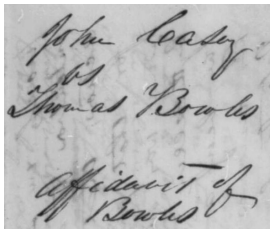
H01



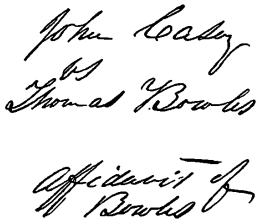
H02



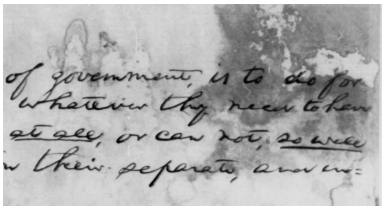
H02



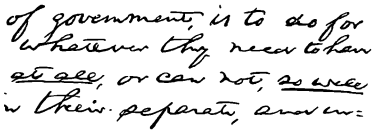
H03



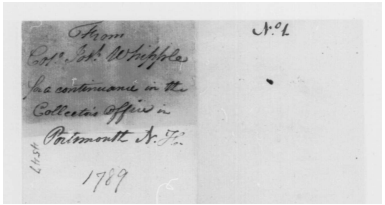
H03



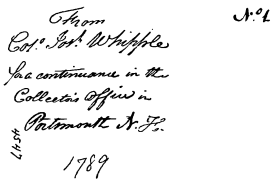
H04



H04



H05



H05

Obrázek B.1: Obrázky v datasetu psaného textu, který je v této práci využit. Vlevo vstupní obrázek, vpravo odpovídající vyprahovaný výsledek.

Příloha C

Obsah paměťového média

Přiložené paměťové médium obsahuje následující:

- `thesis_tex/` – Tato složka obsahuje veškeré zdrojové soubory tohoto dokumentu
- `dataset/` – Tato složka obsahuje obrázky psaného textu, jež byly využity v evolučních algoritmech této práce.
- `cgp/` – Tato složka obsahuje program `cgp`.
  - `src/` – Tato složka obsahuje zdrojové soubory.
  - `dataset/` – Tato složka obsahuje dataset pro evoluci.
  - `Makefile` – Soubor pro překlad.
  - `README.txt` – Soubor s popisem programu.
- `nsga2/` – Tato složka obsahuje program `nsga2`.
  - `src/` – Tato složka obsahuje zdrojové soubory.
  - `dataset/` – Tato složka obsahuje dataset pro evoluci.
  - `input_data/` – Tato složka obsahuje soubor `threshold.in` definující parametry evoluce.
  - `Makefile` – Soubor pro překlad.
  - `README.txt` – Soubor s popisem programu.