VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

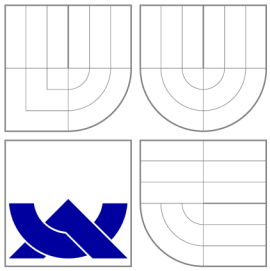
OPTIMALIZACE TVARU VÝFUKOVÝCH SVODŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

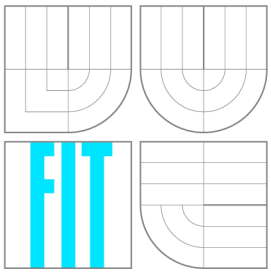
AUTOR PRÁCE
AUTHOR

Bc. DUŠAN NAVRÁTIL

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OPTIMALIZACE TVARU VÝFUKOVÝCH SVODŮ

OPTIMISATION OF EXHAUST DRAINS SHAPE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. DUŠAN NAVRÁTIL
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing. FRANTIŠEK VÍTĚZSLAV ZBOŘIL, CSc.
SUPERVISOR

BRNO 2011

Abstrakt

V této práci je vyvinut systém pro multikriteriální optimalizaci tvaru výfukových svodů včetně počátečního návrhu svodů. Prostor řešení je prohledáván na základě evolučních algoritmů. Ohodnocení tvaru výfukových svodů vychází z délky svodů a součtu obloukových úhlů. Zároveň svody nesmí zasahovat do okolních dílů. Systém je otestován na sadě vstupních dat vycházejících z praxe. Dále je vyhodnocena výkonnost navrženého evolučního algoritmu.

Abstract

Multiobjective optimization system of exhaust manifold shapes including initial design has been developed. Space of possible solutions is explored by an evolutionary algorithm. Evaluation of exhaust drains shape comes from drains length and sum of arc angles. Drains mustn't interfere in surrounding parts. System is tested on set of input data originated from practice. Further, performance of proposed evolutionary algorithm is evaluated.

Klíčová slova

výfukové svody, automatický návrh, multikriteriální optimalizace, evoluční algoritmy.

Keywords

exhaust drains, automated design, multiobjective optimization, evolutionary algorithms.

Citace

Dušan Navrátil: Optimalizace tvaru výfukových svodů, diplomová práce, Brno, FIT VUT v Brně, 2011

Optimalizace tvaru výfukových svodů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Františka Zbořila. Použil jsem pouze podklady uvedené v přiloženém seznamu

.....
Dušan Navrátil
18. května 2011

Poděkování

Děkuji vedoucímu práce panu Františku Zbořilovi za cenné připomínky a odbornou pomoc. Také děkuji konzultantovi Vítovi Pippalovi za ochotné zodpovězení všech mých dotazů k problematice. Dále bych rád poděkoval mé rodině a mým blízkým za velkou podporu.

© Dušan Navrátil, 2011.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **3**

2 Multikriteriální optimalizace **5**
2.1 Základní principy 5
2.2 Multikritériální evoluční algoritmy 6
2.2.1 Způsob přiřazení fitness funkce 7
2.2.2 Zajištění diverzity 8
2.2.3 Obnova populace 9
2.2.4 MOEA s omezujícími podmínkami 9
2.3 NSGA-II 11
2.3.1 Řazení jedinců na základě dominance 11
2.3.2 Zajištění diversity 12
2.3.3 Hlavní smyčka algoritmu 13

3 Analýza úlohy **15**
3.1 Vstupní parametry úlohy 16
3.2 Řešení problému 17
3.3 Omezující podmínky 18
3.4 Kritéria optimalizace 20
3.5 Úloha z pohledu optimalizace 21
3.6 Požadavky na implementaci 22

4 Návrh řešení **23**
4.1 Zakódování jedince 24
4.2 Výpočet a začlenění omezujících podmínek 27
4.3 Variační operátory 31
4.3.1 Křížení 32
4.3.2 Mutace 32
4.4 Generování počáteční populace 34
4.4.1 Generování trubky 34
4.4.2 Evoluce počáteční populace pro NSGA-II 34

5 Implementace a testování systému **36**
5.1 Implementace systému 36
5.1.1 Implementace analytických výpočtů 37
5.1.2 Implementace evolučního algoritmu 40
5.2 Výběr testovací sady 40
5.3 Měření výkonnosti MOEA 42

5.3.1 Míra S pro konvergenci . . . 43
5.3.2 Míra E pro diverzitu aproximace Pareto fronty . . . 44
5.3.3 Míra F pro fenotypovou diverzitu . . . 44
5.3.4 Míry C a D pro porovnání množin řešení . . . 45
5.4 Odhad řídících parametrů . . . 46
5.4.1 Generování počáteční populace pomocí RCGA . . . 46
5.4.2 Multikriteriální optimalizace pomocí NSGA-II . . . 49
5.5 Dosažené výsledky . . . 51

6 Experimenty s nastavením evolučního algoritmu 54
6.1 Experimenty se začleněním omezujících podmínek . . . 54
6.1.1 Tolerance velikosti středových úhlů . . . 54
6.1.2 Tolerance nedodrženého počtu ohybů . . . 54
6.2 Experimenty s generováním nové populace . . . 55
6.2.1 Zvýšení populace potomků . . . 55
6.2.2 Mutace s adaptačními odchylkami normálního rozložení . . . 56
6.2.3 Mutace na základě pořadí vrcholů . . . 57
6.3 Dosažené výsledky . . . 58

7 Kritické části optimalizace 62

8 Závěr 64

A Dosažené aproximace Pareto fronty 68

B Výsledky měření pro jednotlivé varianty NSGA-II 72

Kapitola 1

Úvod

Výfukový systém odvádí od motoru plyny vznikající při spalování směsi paliva. Hladkým odvodem výfukových plynů je dosahováno vyššího výkonu vozu. Kromě toho má výfuk přímý vliv na hlučnost a redukci emisí a navíc chrání zdraví pasažérů tím, že odvádí nebezpečné emisní plyny až za vozidlo.

Výfukové svody odvádějí výfukové plyny přímo od hlav válců motoru. Návrh tvaru výfukových svodů se často tvoří pomocí CAD[1] nástrojů metodou pokusů a omylů. Návrh bývá dost komplikovaný, protože je na něj kladeno mnoho požadavků. Je spojen s mnoha experimenty a analýzami. Automatický návrh by mohl redukovat technické, plánovací a cenová rizika vývoje nového motoru.

Cílem této práce je navrhnout a implementovat systém pro automatický návrh výfukového potrubí podle více kritérií. Kritéria optimalizace vycházejí ze zkušeností odborníků v oblasti návrhu a výroby výfukových systémů a jsou postaveny na geometrických vlastnostech výfukových svodů. V této úloze se pod pojmem výfukový systém omezíme pouze na kolekci trubek vedoucí z motoru do kolektoru výfuku. To bývá často jedna z obtížných částí při návrhu výfukového systému.

Druhá kapitola obsahuje definici obecného multikriteriálního optimalizačního problému podle [23] a stěžejní pojmy v multikriteriální optimalizaci. Dále se zabývá multikriteriálními evolučními algoritmy (dále pouze MOEA[2]) a jejich návrhem, protože právě tuto optimalizační metodu použijeme pro řešení úlohy. Podrobně se také zaměříme na jednoho ze zástupců těchto algoritmů, NSGA-II[3].

V třetí kapitole provedeme analýzu úlohy. Popíšeme omezující podmínky úlohy, které vychází z konstrukčních a výrobních požadavků na výfukové svody. Představíme jednotlivá optimalizační kritéria. Vymezíme návrhové parametry optimalizace a jejich vztahy k vstupním parametrům úlohy. Na konci kapitoly se budeme zabývat předpokládanými problémy z hlediska optimalizace a požadavky na implementaci systému.

Ve čtvrté kapitole navrhneme MOEA pro optimalizaci výfukových svodů. Návrh vychází z již existujícího algoritmu NSGA-II, který je přizpůsoben pro řešenou úlohu. Jednotlivé části algoritmu budou podrobně vysvětleny.

V páté kapitole navržený systém implementujeme a otestujeme na sadě úloh vycházejících z praxe. Zároveň je představen způsob měření výkonnosti MOEA. Kapitola se také zabývá odhadem řídících parametrů navrženého evolučního algoritmu a v závěru jsou zhodnoceny dosažené výsledky.

[1] Computer Aided Design
[2] Multi-Objective Evolutionary Algorithm
[3] Non-dominated Sort Genetic Algorithm

S evolučním algoritmem dále experimentujeme v šesté kapitole. Je představeno pět experimentů s nastavením optimalizace a nové varianty jsou opět zhodnoceny na sadě testovaných úloh.

Kritické části optimalizační metody analyzujeme v sedmé kapitole. Zároveň nastíníme možná vylepšení metody.

Nakonec zhodnotíme výsledky dosažené vybranou metodou, jejich použitelnost v praxi, a nastíníme možná pokračování projektu.

Kapitola 2

Multikriteriální optimalizace

2.1 Základní principy

Předpokládáme libovolný problém s m omezujícími podmínkami a k optimalizačními kritérii, které chceme minimalizovat a všechny považujeme za stejně důležité. Řešení tohoto problému můžeme označit jako *vektor proměnných* $(x_1, x_2, ..., x_n)$ v *prostoru proměnných* $\mathbf{X}$. Proměnným se také říká *návrhové parametry*. Omezující podmínky jsou vyjádřeny pomocí funkcí $g_i : \mathbf{X} \to \mathbb{R}$ a pro přípustné řešení musí platit $g_i(\mathbf{x}) \leq 0, \forall i = 1, ..., m$. Funkce $\mathbf{f} : \mathbf{X} \to \mathbf{Y}$ přiřazuje specifickému řešení *vektor kritérií* $(y_1, y_2, ..., y_k)$ z prostoru kritérií $\mathbf{Y}$ a tím ohodnocuje jeho kvalitu.

Nyní předpokládejme, že vektor kritérií je vektorem reálných čísel, tzn. $\mathbf{Y} \subseteq \mathbb{R}$ a minimalizujeme pouze jedno kritérium. Řešení $\mathbf{x}^1 \in \mathbf{X}$ je lepší než jiné řešení $\mathbf{x}^2 \in \mathbf{X}$, když $y^1 < y^2$, kde $y^1 = f(\mathbf{x}^1)$ a $y^2 = f(\mathbf{x}^2)$. Přestože může v prostoru proměnných existovat více optimálních řešení, existuje pouze jedno optimum v prostoru kritérií.

V případě více-kriteriální funkce $\mathbf{f}$, tedy $\mathbf{Y} \subseteq \mathbb{R}^k$ pro $k > 1$, je porovnávání dvou řešení $\mathbf{x}^1$ a $\mathbf{x}^2$ komplexnější a využívá dobře známého konceptu *Pareto dominance*.

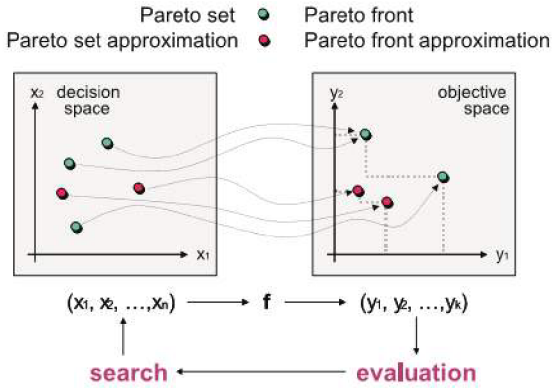
Definice 2.1.1 *Říkame, že vektor* $\mathbf{y}^1$ *dominuje vektor* $\mathbf{y}^2$*, když* $\mathbf{y}^1$ *je nejméně tak dobré jako* $\mathbf{y}^2$ *pro všechna kritéria, a* $\mathbf{y}^1$ *je ostře lepší než* $\mathbf{y}^2$ *alespoň pro jedno kritérium:*

$$\mathbf{y}^1 \prec \mathbf{y}^2 \Leftrightarrow \forall i : \mathbf{y}_i^1 \leq \mathbf{y}_i^2 \wedge \exists i : \mathbf{y}_i^1 < \mathbf{y}_i^2 \quad i = 1, ..., k \tag{2.1}$$

Podobně můžeme tvrdit, že vektor $\mathbf{x}^1$ *dominuje* vektor $\mathbf{x}^2$, když $\mathbf{f}(\mathbf{x}^1)$ *dominuje* $\mathbf{f}(\mathbf{x}^2)$. Množina optimálních řešení, tj. řešení nedominované žádným jiným řešením, může být reprezentována různými vektory v prostoru kritérií. To znamená, že zde může být několik vektorů kritérií, které vyjadřují různý kompromis mezi kritérii.

Definice 2.1.2 *Vektor proměnných* $\mathbf{x}^* \in \mathbf{X}$ *nazýváme Pareto optimem, pokud neexistuje žádný jiný vektor* $\mathbf{x} \in \mathbf{X}$*, který jej dominuje.*

Množinu optimálních řešení v prostoru proměnných $\mathbf{X}$ obecně označujeme jako *Pareto množinu* $\mathbf{X}^* \subseteq \mathbf{X}$ a její obraz do prostoru kritérií $\mathbf{Y}$ označujeme jako *Pareto frontu* $\mathbf{Y}^* = \mathbf{f}(\mathbf{X}^*) \subseteq \mathbf{Y}$. Znalost této množiny potom pomáhá určit nejlepší kompromis mezi kritérii. Cílem optimalizace je nalézt *aproximaci Pareto množiny* – množina vzájemně nedominovaných řešení.



Obrázek 2.1: Ilustrace obecného problému multikritériální optimalizace (převzato z [23])

2.2 Multikritériální evoluční algoritmy

Evoluční algoritmy (EA) jsou stochastické optimalizační metody, které simulují proces přirozené evoluce. Patří mezi ně hlavně genetické algoritmy, evoluční programování a evoluční strategie. Všechny tyto přístupy pracují se sadou kandidátních řešení, která je dále modifikována zejména pomocí dvou stěžejních principů: výběrem a obměnou kandidátů [3].

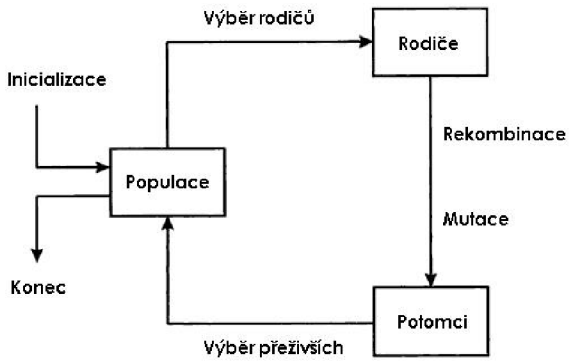
Analogicky s přirozenou evolucí je kandidátní řešení nazýváno *jedincem* a množina kandidátních řešení je označována jako *populace*. Každý jedinec reprezentuje možné řešení, tedy vektor proměnných, avšak není tímto vektorem ale je zakódován vhodnou reprezentací. Kódovanou reprezentaci nazývame *genotypem*, kdežto příslušný vektor proměnných nazýváme *fenotypem*.

Proces výběru rodičovských jedinců se obvykle skládá ze dvou fází: Přiřazení fitness a vzorkování populace. V první fázi jsou jedinci vyhodnoceni v prostoru kritérií a poté je jim přiřazena skalární hodnota, *fitness*, odrážející jejich kvalitu. Na základě této hodnoty jsou poté náhodně vybíráni jedinci určeni k reprodukci. Často používanou metodou je selekce turnajem, kdy je náhodně zvolen určitý počet jedinců z populace (většinou dva) a jedinec s lepší fitness je zkopírován do rodičovské populace, místa, kde jsou shromážděni jedinci určeni k reprodukci. Existuje ale mnoho technik jak výběr provést [20].

Poté jsou na jedince v rodičovské populaci aplikovány variační operátory. Obvykle máme dva typy těchto operátorů: rekombinační a mutační. Rekombinační operátor vezme určitý počet rodičů a vytvoří předdefinovaný počet potomků kombinováním úseků rodičů. To se děje na základě pravděpodobnosti křížení. Na rozdíl od rekombinace, mutace většinou modifikuje jedince pouze z malé části na základě určité pravděpodobnosti.

Nakonec je podle způsobu obnovy populace rozhodnuto, kteří jedinci zůstanou v populaci pro následující generaci. Jedním ze způsobů je nahrazení celé populace potomky. Dalším je například sloučení původní populace s populací potomků do jedné a výběr na základě nějakého deterministického postupu.

Simulace přirozené evoluce je provedena pomocí iteračního výpočtu, jehož obecné schéma je znázorněno na obrázku 2.2. Nejdříve se tedy náhodně vygeneruje počáteční populace. Následuje smyčka, kde je provedeno ohodnocení, výběr, rekombinace a/nebo mutace jedinců. Každá iterace tohoto výpočtu je nazývána *generací*. Podmínka ukončení algoritmu bývá často definována maximálním počtem generací. Jinou podmínkou může být stagnace popu-



Obrázek 2.2: Obecné schéma evolučního algoritmu

lace či existence jedince s dostatečným ohodnocením [6].

Přestože jsou tyto principy jednoduché, evoluční algoritmy prokázaly, že jsou obecnými, robustními a výkonnými prohledávacími algoritmy. Jejich charakteristiky jsou vhodné pro problémy s mnoha konfliktními cíli a také pro problémy s nezvladatelně velkými a vysoce komplexními prohledávacími prostory. Bylo navrženo mnoho variant těchto algoritmů a v poslední době je zaznamenán vzrůstající zájem v oblasti multikriteriálních evolučních algoritmů [23].

Cílem MOEA je aproximovat Pareto množinu řešení. Stěžejními principy návrhu MOEA je navádět prohledávání směrem k Pareto množině (konvergence) a udržovat rozmanitou množinu nedominovaných řešení (diverzita).

První cíl je vázán zejména na výběr rodičovských jedinců, konkrétně v přiřazení skalárních fitness hodnot vzhledem k existenci více kritérií. Druhý cíl se zabývá výběrem z širšího pohledu, protože nechceme, aby populace obsahovala většinu identických řešení. Dalším předmětem návrhu algoritmu je otázka obnovy populace, tedy jak předcházet potenciální ztrátě nedominovaných řešení.

Na každý z těchto aspektů MOEA se zaměříme podrobněji: přiřazení fitness, zajištění diverzity a způsob obnovy populace.

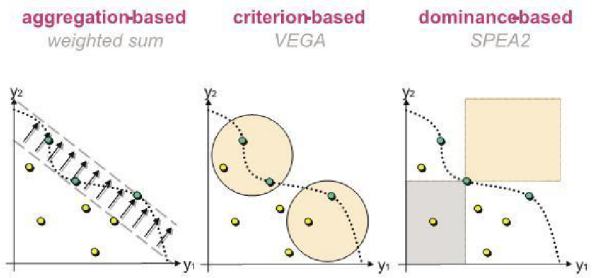
Pokud se prostor návrhových parametrů pohybuje v omezeném prostoru, musíme se zabývat otázkou, jak vhodně začlenit omezující podmínky do algoritmu. Podíváme se na odzkoušené přístupy, jak tento problém řešit.

2.2.1 Způsob přiřazení fitness funkce

Obecně můžeme rozlišit tři strategie přiřazení fitness:

Metody založené na agregaci vychází z tradičních technik a jejich princip spočívá v agregaci kritérií do jedné parametrizované funkce. Parametry jsou měněny během evoluce, abychom dostali více kompromisních řešení. Některé MOEA používají vážený součet kritérií, kde váhy jsou právě těmito parametry.

Metody založené na kritériu přepínají mezi optimalizačními kritérii během fáze výběru. Pokaždé, když jsou vybíráni jedinci pro reprodukci, různé kritérium může rozhodovat o tom, který člen populace bude rodičovským jedincem.



Obrázek 2.3: Různé způsoby ohodnocení jedince (převzato z [23])

Jiné formy tohoto přístupu používají více jednokriteriálních algoritmů, které si po určitém počtu iterací vyměňují genetickou informaci populace. Narozdíl od klasických operátorů v GA, tyto algoritmy pracují s tzv. *informovanými* operátory, které generují více jedinců a na základě aproximace fitness se zvolí pouze jeden potomek. Důležité ale je, že aproximace fitness v informovaném operátoru používá k výpočtu jiné kritérium než je použita k selekci rodičovských jedinců [4].

Metody založené na dominanci ohodnocují jedince pomocí již zmíněného Pareto konceptu. Některé přístupy používají k určení fitness pořadí dominance, tj. počet jedinců, kterými je jedinec dominován. Jiné přístupy využívají hloubky dominance. Populace je rozdělena do několika front a hloubka označuje frontu, do které jedinec patří. Jedinci uvnitř fronty jsou navzájem nedominovaní. Alternativou může být počet dominancí, tj. počet jedinců, kteří jsou dominováni určitým jedincem. Například SPEA[1] nebo SPEA2 využívají obou těchto principů: hloubka a počet dominancí. Přístupy založené na dominanci vychází z celé populace na rozdíl od metod založených na agregaci, kde jedinec je ohodnocen nezávisle od dalších jedinců v populaci.

2.2.2 Zajištění diverzity

Mnoho MOEA se snaží udržovat diverzitu aproximace Pareto množiny začleněním informace o hustotě jedinců v prostoru kritérií do procesu výběru rodičovských jedinců. Čím větší hustota v okolí jedince, tím menší je jeho šance být vybrán. Problém je úzce spojen s odhadem funkce hustoty pravděpodobnosti ze statistiky a metody jsou tedy podle toho rozděleny do kategorií.

Metody založené na jádře definují okolí bodu pomocí funkce K, která vezme vzdálenost k jinému bodu jako argument. V praxi se pro každého jedince vypočítají vzdálenosti d_i k ostatním jedincům i a po aplikování funkce K se sečtou výsledné hodnoty $K(d_i)$. Tato suma potom reprezentuje odhad hustoty pro určitého jedince.

Sdílení fitness je nejvíce popularní technika tohoto typu metod v evolučních algoritmech a je použita například v MOGA[2] nebo NSGA.

Metody založené na nejbližším sousedovi používají vzdálenost jedince ke k-tému nejbližšímu sousedovi k odhadu hustoty v okolí jedince. Odhadem je obvykle funkce in-

[1] Strength Pareto Evolutionary Algorithm
[2] Multi-objective Genetic Algorithm

verzní ke vzdálenosti. Například SPEA2 vypočítá vzdálenost ke k-tému nejbližšímu jedinci a převrácenou hodnotu přičte k fitness (uvažujeme minimalizaci fitness).

Metody založené na mřížce definují pomocí hypermřížky sousedství v prostoru. Hustota kolem jedince je potom jednoduše určena počtem jedinců nacházejících se ve stejné buňce mřížky. Mřížka může být pevná, nebo může být adaptovaná pro aktuální populaci.

Poznamenejme, že všechny tyto metody vyžadují vzdálenostní funkci, která může být definována na genotypu, fenotypu s ohledem na prostor proměnných nebo fenotypu s ohledem na prostor kritérií. Většína přístupů uvažuje vzdálenost mezi jedinci jako vzdálenost mezi vektory kritérií.

2.2.3 Obnova populace

Způsob obnovy populace je spojen se ztrátou dobrých řešení zapříčiněnou stochastickými efekty. Jedním ze způsobů je sloučení staré populace spolu s potomky a uplatnění deterministické procedury. Ten druhý udržuje sekundární populaci, tzv. archiv, a v každé generaci do něj vkládá slibné jedince. Tento archiv může sloužit jako externí uložistě oddělené od optimalizačního cyklu anebo může být integrován do EA tak, že jeho členové se účastní fáze výběru.

Nejčastěji se jedinci porovnávají na základě dominace. Pokud je udržován archiv, obsahuje pouze aktuální aproximaci Pareto fronty. To znamená, že všichni dominovaní jedinci jsou z něj vyloučeni. Avšak obnova na základě dominance není obecně dostačujícím řešením, takže k obnově populace bývá použita dotatečná informace. Příklady jsou informace o hustotě a čas (v generacích), který uplynul od doby, kdy jedinec vstoupil do archivu.

Většina elitistických MOEA používá kombinaci dominace a hustoty k výběru jedinců, kteří setrvají v archivu. Avšak tyto principy mohou trpět problémem úpadku, kdy jedinci přítomni v archivu v generaci t mohou být dominováni jedinci, kteří byli členy archivu v generaci $t' < t$ a byli později odstraněni. V poslední době byly prezentovány metody, jak se vyhnout tomuto problému a udržovat rozmanitou množinu Pareto optimálních řešení [23, 13].

V [21] je vyhodnocena výkonnost různých multikritériálních evolučních algoritmů na sadě testovacích úloh. Studie mimojiné diskutuje příznivé účinky elitismu na konvergenci MOEA.

2.2.4 MOEA s omezujícími podmínkami

V praxi se vyskytuje celá řada multikritériálních problémů s omezujícími podmínkami. Avšak při návrhu nových MOEA se na tuto skutečnost často zapomínalo. Proto se v poslední době vyvíjejí různé techniky, jak omezující podmínky zakomponovat do evolučních algoritmů. Připadně vznikají nové verze již známých algoritmů jako například NSGA-II [5].

Obecně je potřeba nalézt algoritmus, který se dokáže vypořádat s omezujícími podmínkami a efektivně určit Pareto optimální množinu. U genetických algoritmů se často používají penalizační techniky, kdy ke každé kritérialní funkci přičteme penalizační hodnotu reflektující porušení omezujících podmínek. Fitness funkce $\mathbf{\Psi}(\mathbf{y}) = (\psi_1(\mathbf{y}), \psi_2(\mathbf{y}), ..., \psi_k(\mathbf{y}))$, která transformuje definici optimalizačního problému s omezujícími podmínkami (uvedenou v 2.1) do problému bez omezení, vypadá následovně:

$$\psi_i = f_i(\mathbf{x}) + r_g \cdot \phi(g_j(\mathbf{x})) \quad j = 1, ..., m \quad i = 1, ..., k \quad \phi \geq 0 \tag{2.2}$$

ϕ je funkce ukládající pokutu na základě posloupnosti *penalizačních parametrů*.

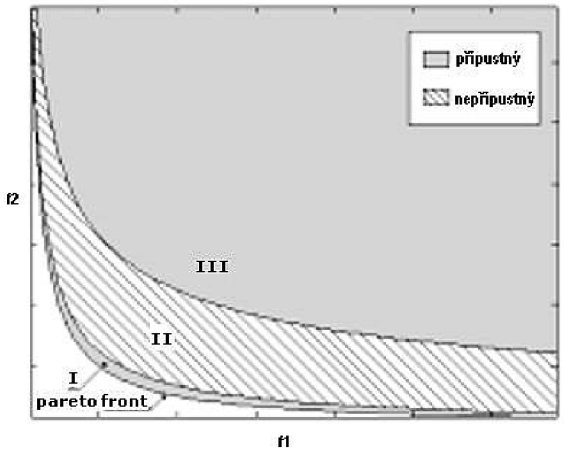
Tento klasický přístup ovšem obsahuje určité slabiny, protože není znám vhodný penalizační parametr r_g. Příliš vysoké hodnoty nezvýrazňují optimalizační kritéria a většinu úsilí potom algoritmus tráví hledáním přípustných jedinců. Nízké hodnoty zase navádějí směr prohledávání k nepřípustným jedincům.

Problém nalezení vhodných penalizačních parametrů se snaží řešit algoritmus PP-NRGA[3] [10]. Uvedený penalizační parametr r_g balancuje mezi kritérii a penalizační funkcí. Modifikace fitness funkce vypadá následovně:

$$\psi_i(\mathbf{x}) = f_i(\mathbf{x}) + rank_{f_i} + rank_g \sum_{j=1}^{m} \phi(g_j(\mathbf{x})) \tag{2.3}$$

$rank_{f_i}$ je pořadí jedince na základě i-tého kritéria v intervalu $\langle 1, N\rangle$, kde N je velikost populace, a $rank_g$ je pořadí na základě součtu funkcí ohodnocující porušení omezujících podmínek v intervalu $\langle N+1, 2\cdot N\rangle$.

Odlišnou technikou je začlenění omezujících podmínek jako kritéria optimalizace. Při obnově populace se uplatňují speciální metody založené na elitismu, které zohledňují porušení omezujících podmínek. Tento přístup se snaží řešit problém znázorněn na obrázku 2.4. Oblast III je největší, obsahuje přípustné jedince a počáteční populace bude s největší pravděpodobností tvořena jedinci právě z této oblasti, zatímco je dost nepravděpodobné, že z oblasti I bude vygenerován nějaký jedinec. Použitím klasické penalizační techniky budou jedinci z oblasti II dominováni těmi z oblasti III, protože kritéria jedinců patřících do oblasti II budou navýšena o penalizační hodnoty. To vrátí směr prohledávání zpátky do oblasti III. Oblast II tedy tvoří jakousi „zeď“ mezi oblastmi I a III. Naopak při začlenění omezujících podmínek jako kritéria není oblast bariérou, protože jedinci z oblasti II nejsou dominováni těmi z III [18]. Tato technika přesto generuje velké množství nepřípustných jedinců a tak se může stát, že prohledávání prostoru povede nežádoucím směrem. Proto jsou při obnově populace vyjmuti nepřístupní jedinci blízcí jiným přípustným jedincům na základě vzdálenosti v prostoru kritérií.



Obrázek 2.4: Hypotetický problém (převzato z [18])

[3]Parameterless Penalty Non-dominated Ranking Genetic Algorithm

Dalším často používaným způsobem je začlenění omezujících podmínek do operátoru selekce rodičovských jedinců. Pro představu zmíníme navrženou modifikaci turnajového výběru, kdy jsou z populace náhodně vybráni dva jedinci za účelem stanovit vítěze turnaje, příštího rodiče. Princip je následující:

1. Pokud jsou oba jedinci přípustní, potom jsou porovnáváni k přípustným členům porovnávací množiny, která obsahuje specifický počet náhodně vybraných jedinců z populace. Pokud jeden z nich je nedominovaný v této množině a druhý je dominovaný, vítězem se stává nedominovaný jedinec. Pokud jsou oba dominovaní nebo nedominovaní, o vítězství rozhodne zvolená funkce odhadující hustotu v prostoru kritérii.

2. Pokud je jeden z nich přípustný a druhý nepřípustný, vítězem turnaje je přípustný jedinec.

3. Pokud ani jeden z nich není přípustný, jsou porovnáváni k nepřípustným členům porovnávací množiny na základě porušení omezujících podmínek. Pokud je jeden z nich lepší než nejlepší nepřípustný jedinec z porovnávací množiny a druhý je horší než tento nejlepší jedinec, vybere se ten první. Pokud jsou oba lepší nebo horší než nejlepší nepřípustný jedinec, o vítězství rozhodne zvolená funkce odhadující hustotu v okolí jedince.

Výhodou tohoto přístupu je, že nepřípustné jedince nevyhodnocujeme v prostoru kritérií, což může znatelně ušetřit celkový výpočetní čas.

Autoři této techniky uvedli také modifikaci pro *ranking* selekci, kdy je populace seřazena podle dominance a diverzity a nepřípustné jedince přesune na konec populace. Při výběru rodičovských jedinců je pro každého jedince $\mathbf{x}$ určena pravděpodobnost výběru $Prob(\mathbf{x})$ na základě jeho pořadí $rank(\mathbf{x})$ v seřazené populaci a parametru rozložení pravděpodobnosti q [11]:

$$Prob(\mathbf{x}) = q \cdot (1-q)^{rank(\mathbf{x})-1} \tag{2.4}$$

Při návrhu začlenění omezujících podmínek je také možné se inspirovat technikami, jak převést SOEA[4] s omezujícími podmínkami na MOEA bez omezujících podmínek [14], případně dalšími návrhy zohledňující přípustnost řešení při vyhodnocení fitness [8].

2.3 NSGA-II

Deb a spol. v [5] představili evoluční algoritmus NSGA-II, který se poté stal jedním se zástupců MOEA, vedle dalšího často používaného algoritmu SPEA [22]. Snaží se odstranit nedostatky jeho předchůdce NSGA, zejména snížit výpočetní náročnost algoritmu. NSGA-II byl již použit k řešení mnoha praktických optimalizačních problémů (např. [9], [1]).

Následuje podrobný popis jednotlivých částí algoritmu. Poté je zapojíme do hlavního cyklu algoritmu, kde taktéž rozebereme složitost jednotlivých částí.

2.3.1 Řazení jedinců na základě dominance

NSGA-II přiřazuje jedinci fitness na základě úrovně dominance. Populaci tedy potřebujeme rozdělit do množin tak, že uvnitř množiny se jedinci vzájemně nedominují. Abychom nalezli množinu nedominovaných řešení – aproximaci Pareto množiny, každý jedinec musí být porovnán s ostatními a prověřen, zda jej některý z nich nedominuje. Algoritmus 1 ukazuje

[4]Single-Objective Evolutionary Algorithm - jednokriteriální evoluční algoritmus

strategii, kterou používá NSGA-II pro nalezení nedominované fronty v populaci. Každý jedinec p je porovnáván s částečně vyplněnou populací P'. Pokud tento jedinec dominuje některého jedince $q \in P'$, jedinec q je z populace P' vyřazen. Pokud naopak některý jedinec q dominuje jedince p, p nebude zahrnut do populace P'. Pokud ale žádný jedinec q nedominuje p, p je vložen do populace P'. Na konci je populace P' prohlášena za hledanou množinu nedominovaných řešení.

Z uvedeného algoritmu lze vidět, že druhý jedinec z populace je porovnán maximálně s jedním řešením, třetí jedinec maximálně se dvěma jedinci, atd. Nalezení první nedominované fronty má potom celkovou složitost $O(MN^2)$, kde M je počet kritérií a N je velikost populace. Odhad složitosti byl experimentálně potvrzen v [5].

Algoritmus 1 $P' =$ `find-non-dominated-front`(P) [5]

Vstup: P Populace jedinců
Výstup: P' Množina nedominovaných jedinců

```
1:  P' ← {1}
2:  for all p ∈ P ∧ p ∋ P' do
3:     P' ← P' ∪ {p}
4:     for all q ∈ P' ∧ q ≠ p do
5:        if p ≺ q then
6:           P' ← P' \ {q}
7:        else if q ≺ p then
8:           P' ← P' \ {p}
9:        end if
10:    end for
11: end for
```

Pro nalezení dalších front odebereme jedince v P' z populace P a proces nalezení fronty se opakuje, jak je vidět v algoritmu 2. Přitom se zachovává složitost v nejhorším případě $O(MN^2)$. Na konci získáme fronty $\mathcal{F}_1, \mathcal{F}_2, ...$, jež představují jednotlivé úrovně dominance.

Algoritmus 2 $\mathcal{F} =$ `non-dominated-sort`(P) [5]

Vstup: P, Populace jedinců
Výstup: $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, ...\}$, Množina nedominovaných front

```
1: i ← 1
2: while P ≠ ∅ do
3:    F_i ← find-non-dominated-front(P)
4:    P ← P \ F_i
5:    i ← i + 1
6: end while
```

2.3.2 Zajištění diversity

Původní NSGA používá princip *sdílené fitness*, jež vyžaduje od uživatele zadání parametru vyjadřujícího rozsah sdílení. Tento přístup byl v NSGA-II nahrazen *operátorem přemnožení*, který nevyžaduje žádný parametr pro zajištění diverzity v populaci.

Odhad hustoty populace kolem určitého jedince v populaci počítáme na základě prostoru kritérií. Pro výpočet této vzdálenosti musíme vzestupně seřadit populaci podle každého kritéria. Krajním jedincům přiřadíme jako vzdálenost nekonečno. Vzdálenost ostatních jedinců

vypočítáme jako absolutní rozdíl funkčních hodnot dvou sousedních jedinců. Celková vzdálenost jedince je určena jako součet vzdáleností pro příslušná kritéria.

Algoritmus 3 popisuje výpočet vzdálenosti pro množinu nedominovaných jedinců $\mathcal{I}$. $\mathcal{I}[i].m$ reprezentuje m-té kritérium i-tého jedince v populaci $\mathcal{I}$. Jelikož musíme provést M-krát seřazení populace o N jedincích, algoritmus má celkovou složitost $O(MN \log N)$.

Algoritmus 3 `distance-assignment`$(\mathcal{I})$ [5]

Vstup: $\mathcal{I}$ Množina nedominovaných jedinců
Výstup: $\mathcal{I}$ Množina nedominovaných jedinců s přiřazenou vzdáleností

```
1:  l = |I|
2:  for i = 1 to l do
3:      I[i]_dist ← 0
4:  end for
5:  for all kritéria m do
6:      I ← sort(I, m) // seřazení podle kritéria
7:      I[1]_dist ← ∞
8:      I[l]_dist ← ∞
9:      for i = 2 to l − 1 do
10:         I[i]_dist ← I[i]_dist + (I[i + 1].m − I[i − 1].m)
11:     end for
12: end for
```

Na konci tohoto procesu jsme schopni porovnávat jedince na základě jejich vzájemné polohy. Jedinec s menší hodnotou vzdálenosti se nachází v přemnožené oblasti.

Nyní má každý jedinec i přiřazeny dva atributy:

- i_{dom} Úroveň dominace, příslušnost k frontě $\mathcal{F}_{dom}$.
- i_{dist} Odhad vzdálenosti.

Na základě těchto atributů je definován již zmíněný operátor přemnožení jako částečné uspořádání $\prec_n$:

$$i \prec_n j \iff i_{dom} < j_{dom} \vee (i_{dom} = j_{dom} \wedge i_{dist} > j_{dist})$$

To znamená, že ze dvou jedinců lišících se úrovní dominance preferujeme jedince s nižší úrovní. Pokud jedinci patří do stejné fronty, preferujeme jedince v méně přemnožené oblasti.

2.3.3 Hlavní smyčka algoritmu

Na začátku je náhodně vygenerována počáteční populace P_0. Tato populace je seřazena na základě dominance, takže každý jedinec má poté přiřazenou úroveň dominance. Nejprve se provede selekce turnajem. Po použití rekombinačních a mutačních operátorů vznikne populace potomků Q_0 o velikosti N. Postup v ostatních iteracích je potom odlišný.

Algoritmus 4 popisuje jednu generaci NSGA-II. Nejdříve dochází ke sjednocení aktuální populace P_t s populací potomků Q_t. Nová populace R_t má velikost $2 \cdot N$. Poté je populace R_t seřazena podle úrovně dominance. Do nové populace potomků P_{t+1} budou umístěny všechny fronty tak, aby součet jejich velikostí byl menší než N. Pokud populace ještě stále není naplněna do velikosti N, další fronta bude seřazena na základě operátoru přemnožení $\prec_n$ a nejlepší jedinci z ní budou vybráni k doplnění zbývajícího místa v P_{t+1}. Nová populace P_{t+1}

Algoritmus 4 Generace algoritmu NSGA-II [5]

1: **globals** P_t, Q_t
2: $R_t \leftarrow P_t \cup Q_t$
3: $\mathcal{F} \leftarrow$ `non-dominated-sort`(R_t)
4: $P_{t+1} \leftarrow \varnothing$
5: $i \leftarrow 1$
6: **while** $|P_{t+1}| + |\mathcal{F}_i| \leq N$ **do**
7: `distance-assignment`$(\mathcal{F}_i)$
8: $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i$
9: $i \leftarrow i + 1$
10: **end while**
11: `sort`$(\mathcal{F}_i, \prec_n)$
12: $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
13: $Q_{t+1} \leftarrow$ `new-pop`(P_{t+1})
14: $i \leftarrow i + 1$

je poté základem pro uplatnění operátorů selekce, křížení a mutace, čímž vznikne populace potomků Q_{t+1}. Selekce turnajem používá k porovnávání jedinců operátor přemnožení $\prec_n$.

Nyní se podívejme na složitosti jednotlivých operací v nejhorším možném případě:

1. $O(MN^2)$ pro nedominované seřazení.
2. $O(MN \log N)$ pro přiřazení vzdálenosti.
3. $O(N \log N)$ pro seřazení na základě operátoru přemnožení.

Celková složitost algoritmu v nejhorším případě je tedy $O(MN^2)$. Kritickou částí algoritmu je tedy nedominované řazení.

Začlenění omezujících podmínek Přístup zabudování omezujících podmínek do NSGA-II vychází z technik často používaných v SOEA, které spočívají v modifikaci selekce turnajem. Jak jsme si již dříve zmínili, jedinci jsou při turnaji porovnávání na základě operátoru přemnožení $\prec_n$. NSGA-II jednoduše upraví definici dominance tak, aby docházelo nejdříve k porovnávání na základě porušení omezujících podmínek, až poté na základě dominance.

Definice 2.3.1 *Řešení* **a** *dominuje vůči omezením řešení* **b**, *pokud jedna z následujících podmínek je pravdivá:*

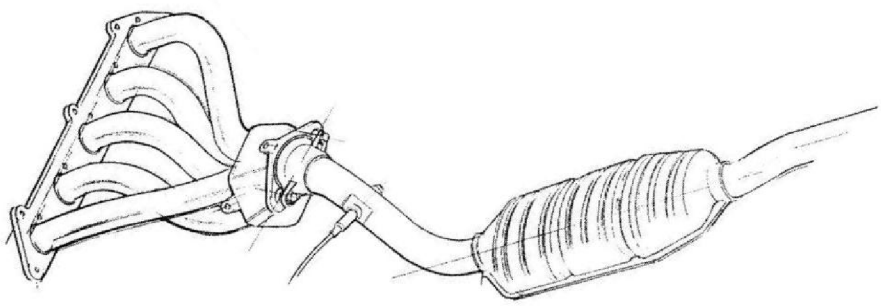
1. *Řešení* **a** *je přípustné a řešení* **b** *není přípustné.*
2. *Ani jedno z řešení není přípustné, ale řešení* **a** *má celkové ohodnocení porušení omezujících podmínek menší než řešení* **b**.
3. *Obě řešení jsou přípustné ale řešení* **a** *dominuje řešení* **b**.

Výsledkem takto zavedeného přístupu je to, že jakýkoli přípustný jedinec má nižší (lepší) úroveň dominance než jakýkoli nepřípustný. Přípustní jedinci jsou seřazeni na základě úrovně dominance a nepřípustní jedinci na základě míry porušení omezujících podmínek. Zavedení této techniky nezmění uvedenou výpočetní složitost algoritmu a nevyžaduje žádné penalizační parametry.

Kapitola 3

Analýza úlohy

Výfukové svody jsou sběrné potrubí, které je pevně uchyceno přímo na hlavu motoru. Tvoří jej několik trubek, odvádějících spaliny od každého válce. Trubky jsou svedeny do jediné pomocí kolektoru výfuku. Tato trubka vyúsťuje v přední výfukové potrubí nebo přímo katalyzátor.



Obrázek 3.1: Výfukový systém osobního automobilu

Systém by měl navrhnout umístění trubek v prostoru. Trubky vedou z válců motoru do kolektoru výfuku. Každá trubka se skládá z posloupnosti rovných úseků a ohybů ve tvaru kruhového oblouku. Úseky musí na sebe tečně navazovat. Trubky musí vyhovovat určitým omezením, která vyplývají z výrobních požadavků na konstrukci trubek (např. velikost středového úhlu oblouku). Dále se nesmí trubky dotýkat či protínat, musí být mezi nimi udržovaná minimální vzdálenost. Trubky musí taktéž vyhovovat prostorovým omezením (nesmí narážet do okolních dílů). Optimalizačních kritérií je hned několik, přitom je problém definovat, které je důležitější než jiné. Chceme, aby trubky měly stejnou nejkratší možnou délku s nejmenším možným součtem středových úhlů.

Odborné práce zabývající se optimalizací výfukových svodů většinou vychází ze simulace proudění plynů v trubkách. Snaží se tak ohodnotit přímý dopad návrhu na funkci motoru a celého výfukového systému. Například Kanazaki a spol. v [12] se snaží zvýšit výkon motoru a zároveň snížit dopad výfukových splodin na životní prostředí. Optimalizačními kritérii je teplota plynů na konci výfukových svodů a účinnost nabíjení. Vychází se z počátečního návrhu, který je dále optimalizován pomocí multikriteriálního GA, konkrétně DRMOGA[1]. Dále systém neuvažuje kolektor výfuku, trubky se postupně spojují v jednu. Trubka nemusí

[1]Divided Range Multi-objective Genetic Algorithm

mít konstantní průřez a práce studuje mimo jiné i vliv optimalizace průměru trubky na kritéria.

V praxi se často využívá optimalizačních aplikací řešící různé třídy návrhových úloh obecně. U těchto aplikací je nutné definovat kritéria a omezení, což může být pro tuto úlohu problematické. Typickým příkladem aplikace je Pro/ENGINEER BMX [16], který je vestavěn do CAD systému. Při optimalizaci se vychází z určitého počátečního návrhu a systém se snaží prohledávat prostor velkého množství možných řešení neinformovanou modifikací původního návrhu. Proces optimalizace tak může být časově náročný v závislosti na kriteriálních funkcích a omezujících podmínkách.

Z pohledu návrhu výfukových svodů se v této práci se soustředíme na dvě výzvy:

Generování počátečního návrhu výfukových svodů. Narozdíl od předešlých pokusů o optimalizaci výfukových svodů, systém navržený v této práci musí vytvořit počáteční návrh, který bude dále optimalizovat podle kritérií. Tato etapa může být dost náročná v závislosti na prostorových omezení, výrobních požadavcích apod.

Nalezení více optimálních návrhů vyjadřujících kompromis mezi požadovanými kritérii. Množina těchto řešení by měla být dostatečně různorodá, aby uživateli umožnila rozhodovat mezi kritérii.

V této kapitole si podrobně analyzujeme úlohu z různých pohledů. Nejdříve uvedeme výčet všech vstupních parametrů, tedy požadované vlastnosti výfukových svodů. Poté uvedeme potřebné matematické vztahy z analytické geometrie, pomocí nichž definujeme omezující podmínky a vymezíme návrhové parametry optimalizace. Zavedené označení proměnných a konstant budeme používat i v následujících kapitolách. Dále představíme optimalizační kritéria včetně jejich matematického vyjádření. Vlastnosti problému shrneme a formulujeme předpokládané problematické části z pohledu optimalizace. Nakonec stanovíme požadavky na implementaci systému.

3.1 Vstupní parametry úlohy

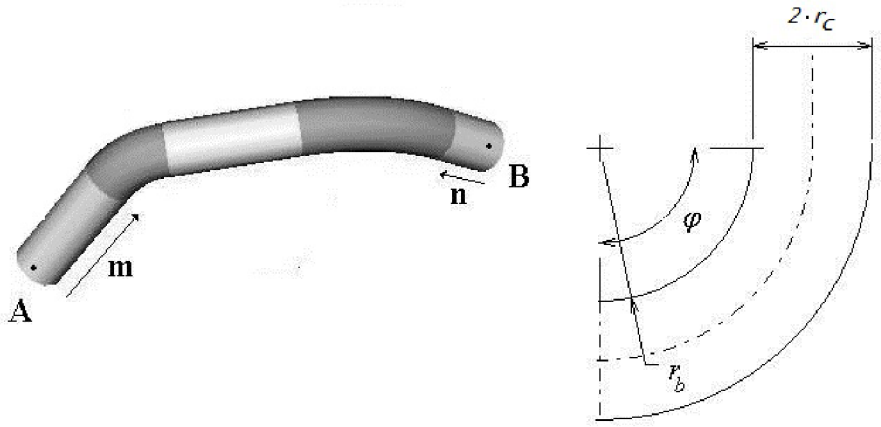
Vstupními parametry úlohy jsou:

- $r_c \in \mathbb{R}, r_c > 0$ Poloměr kruhového průřezu trubky.
- $\varphi_{min} \in \mathbb{R}, 0 < \varphi_{min} < \pi$ Minimální středový úhel ohybu. Uvažujeme pouze konvexní oblouky.
- $r_b \in \mathbb{R}, r_b > 0$ Poloměr ohybu.
- $d_{min} \in \mathbb{R}, d_{min} \geq 0$ Minimální povolená vzdálenost dvou trubek.
- $p \in \mathbb{N}$ Počet trubek.

Pro každou trubku musí být definována šestice parametrů $(\mathbf{A}, \mathbf{B}, \mathbf{m}, \mathbf{n}, c_{min}, c_{max})$ kde $\mathbf{A}, \mathbf{B}$ jsou body a $\mathbf{m}, \mathbf{n}$ jsou vektory v trojrozměrném prostoru. Jejich vztah k definici trubky je znázorněn na obrázku 3.2.

- $\mathbf{A} \in \mathbb{R}^3$ je střed hlavy motoru, odkud musí vycházet osa příslušné trubky.
- $\mathbf{B} \in \mathbb{R}^3$ je střed pozice kolektoru výfuku, kde musí končit osa příslušné trubky.

- $\mathbf{m} \in \mathbb{R}^3, |\mathbf{m}| = 1$ je normálový vektor k rovině hlavy motoru, orientovaný ve směru trubky.
- $\mathbf{n} \in \mathbb{R}^3, |\mathbf{n}| = 1$ je normálový vektor k pozici kolektoru výfuku, orientovaný ve směru trubky.
- $c_{min} \in \mathbb{N}, c_{min} \geq 2$ je minimální počet ohybů.
- $c_{max} \in \mathbb{N}, c_{max} \geq c_{min}$ je maximální počet ohybů.



Obrázek 3.2: Definice začátku a konce trubky

3.2 Řešení problému

Řešením problému je geometrický popis trubek. Jelikož má trubka kruhový průřez, můžeme se omezit pouze na výpočty spojené s osou trubky. Osa se tedy skládá z kruhových oblouků a úseček.

Geometrii konvexního kruhového oblouku můžeme jednoznačně popsat tečnými body $\mathbf{P_1}, \mathbf{P_2}$ a středem $\mathbf{S}$. Dále budeme tyto tři body používat k definici oblouku: $\mathbf{a} = (\mathbf{S}, \mathbf{P_1}, \mathbf{P_2})$ Přitom pro poloměr oblouku r musí platit $r = |\mathbf{P_1 S}| = |\mathbf{P_2 S}| = r_c + r_b$. Platí, že tento poloměr musí být pro všechny oblouky konstantní.

V posloupnosti oblouků a úseček osy trubky je možné, aby dva oblouky následovaly po sobě. Popis trubky můžeme zjednodušit tím, že ji definujeme pouze jako posloupnost oblouků. Pořadí oblouků bude popisovat, jak na sebe jednotlivé oblouky navazují od motoru až po kolektor výfuku.

Trubku $\boldsymbol{t}$ budeme tedy označovat jako konečnou posloupnost oblouků $\mathbf{a}^1, \mathbf{a}^2, ..., \mathbf{a}^c$, kde $c, c_{min} \leq c \leq c_{max}$ je počet oblouků dané trubky. Dva sousední oblouky $\mathbf{a}^i$ a $\mathbf{a}^{i+1}$ jsou potom spojeny úsečkou $\mathbf{P}_2^i \mathbf{P}_1^{i+1}$. Pokud $\mathbf{P}_2^i = \mathbf{P}_1^{i+1}$, oblouky na sebe navazují přímo, přesto budeme v některých případech mezi body uvažovat úsečku nulové délky.

Po takto zavedeném označení řešení úlohy $\mathbf{x}$ představuje posloupnost oblouků:

$$\mathbf{x} = \boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_p = \mathbf{a}_1^1, \mathbf{a}_1^2, ..., \mathbf{a}_1^{c_1}, \mathbf{a}_2^1, \mathbf{a}_2^2, ..., \mathbf{a}_2^{c_2}, ..., \mathbf{a}_p^{c_p} \tag{3.1}$$

Proměnná $c_i, i = 1..p$ je počet oblouků i-té trubky. Pokud chceme zasadit označení řešení do teorie optimalizace jako vektor proměnných, pojem zavedený v kapitole 2.1, stačí rozvinout definici oblouku až na souřadnice jednotlivých bodů:

$$\mathbf{a} = (S_x, S_y, S_z, P_{1x}, P_{1y}, P_{1z}, P_{2x}, P_{2y}, P_{2z}) \tag{3.2}$$

. Takový vektor reálných čísel může mít pro různá řešení různou délku vzhledem k počtu ohybů trubek. Pro jedno konkrétní řešení $\mathbf{x}$ můžeme rozměr tohoto vektoru určit následovně:

$$len(\mathbf{x}) = \sum_{i=1}^{p} 9 \cdot c_i \tag{3.3}$$

Na základě vektoru návrhových parametrů jsme schopni vypočítat kritéria optimalizace a v tomto tvaru je i požadovaným výstupem optimalizace.

3.3 Omezující podmínky

Vycházíme-li z definice optimalizačního problému s omezujícími podmínkami (2.1), musíme pro řešení výfukových svodů $\mathbf{x}$ definovat funkci (nebo více funkcí) $g(\mathbf{x})$, která nám určí, zda řešení $\mathbf{x}$ je proveditelné či nikoli.

$$g(\mathbf{x}) = \begin{cases} \leq 0, & \text{řešení } \mathbf{x} \text{ je proveditelné} \\ > 0, & \text{jinak} \end{cases} \tag{3.4}$$

Máme za cíl pouze stanovit na základě vektoru návrhových parametrů, zda je řešení proveditelné, tj. splňuje požadované omezující podmínky. Při výčtu omezujících podmínek uvedeme pouze vztahy návrhových a vstupních parametrů. Tyto vztahy lze převést na soustavu nerovnic zavedenou při definici optimalizačního problému v kapitole 2.1. Dále budeme využívat zavedené notace z předchozích částí kapitoly.

Omezující podmínky rozdělíme do dvou skupin podle toho, jak se na trubku díváme: konstrukce trubky a interakce trubky v prostoru. U první skupiny nás zajímá konstrukční proveditelnost trubky, zatímco u druhé skupiny nás zajímá pozice trubky v prostoru a možné průniky s dalšími objekty. Tam, kde to bude možné, si uvedeme matematické vyjádření dílčích omezení, u některých vyjádříme omezující podmínky neformálně a budeme se jimi podrobněji zabývat až v návrhu řešení.

Konstrukce trubek Tyto omezující podmínky vycházejí z výrobních požadavků na tvar trubky a definujeme si je pomocí osy trubky. Pro vyjádření vztahů mezi oblouky si pro každý oblouk $\mathbf{a}^i, i = 1..c$ označíme průsečík jeho tečen v bodech $\mathbf{P}_1^i, \mathbf{P}_2^{i+1}$ jako bod $\mathbf{V}^i$. Uvažujeme pouze tečny v rovině oblouku. Tento pomocný bod $\mathbf{V}$ můžeme vypočítat z již zavedené definice konvexního oblouku takto:

$$\begin{aligned} \mathbf{u} &= (\mathbf{P}_1 - \mathbf{S}) \\ \mathbf{v} &= (\mathbf{P}_2 - \mathbf{S}) \\ \mathbf{w} &= \mathbf{u} + \mathbf{v} \\ \mathbf{V} &= \mathbf{S} + t \cdot \mathbf{w} \end{aligned} \tag{3.5}$$

Pro určení bodu $\mathbf{V}$ hledáme parametr t na ose úhlu $\angle \mathbf{P}_1\mathbf{S}\mathbf{P}_2$. Využijeme znalosti goniometrických vzorců, vlastností pravoúhlého trojúhelníka a výpočtu odchylky dvou vektorů:

$$\begin{aligned} \cos(\varphi) &= \frac{\mathbf{u}\cdot\mathbf{v}}{r^2} \\ \cos\left(\frac{\varphi}{2}\right) &= \frac{|\mathbf{v}|}{t\cdot|\mathbf{w}|} \\ |\cos\left(\frac{\varphi}{2}\right)| &= \sqrt{\frac{1+\cos\varphi}{2}} \end{aligned}$$

Po úpravě dostáváme:

$$t = \frac{r}{|\mathbf{w}|\cdot\sqrt{\frac{1+\frac{\mathbf{u}\cdot\mathbf{v}}{r^2}}{2}}} \tag{3.6}$$

Pomocí takto vypočítaných průsečíků $V^1, V^2, ..., V^c$ jsme schopni dále jednoduše vyjádřit omezení pro každý oblouk. Aby bylo možné trubku považovat za konstrukčně proveditelnou, musí splňovat následující podmínky:

- **Středový úhel oblouku**
  Každý středový úhel oblouku musí být větší než povolené minimum φ_{min}. Středový úhel oblouku φ vypočítáme jako odchylku vektorů:

$$\varphi = \arccos\left(\frac{(\mathbf{P}_1-\mathbf{S})\cdot(\mathbf{P}_2-\mathbf{S})}{r^2}\right) \tag{3.7}$$

- **Tečná návaznost oblouků**
  Dva sousedící oblouky musí na sebe tečně navazovat. Pro každé dva oblouky $\mathbf{a}_i, \mathbf{a}_{i+1}, i = 1, .., c-1$ musí platit, že body $\mathbf{P}_2^i, \mathbf{P}_1^{i+1}$ náleží úsečce $\mathbf{V}^i\mathbf{V}^{i+1}$ a bod $\mathbf{V}^i$ musí být blíže bodu $\mathbf{P}_2^i$ než bodu $\mathbf{P}_1^{i+1}$. Podmínku lze vypočítat pomocí parametrického vyjádření přímky. Pro každý bod nalezneme jeho pozici na přímce řešením soustavy rovnic:

$$\mathbf{P}_2^i = \mathbf{V}^i + s\cdot(\mathbf{V}^{i+1}-\mathbf{V}^i) \tag{3.8}$$
$$\mathbf{P}_1^{i+1} = \mathbf{V}^i + t\cdot(\mathbf{V}^{i+1}-\mathbf{V}^i) \tag{3.9}$$

  Potom musí platit: $0 < s \leq t < 1$

  Tato podmínka nám zajistí křivku skládající se z tečně navazujících úseček a kruhových oblouků.

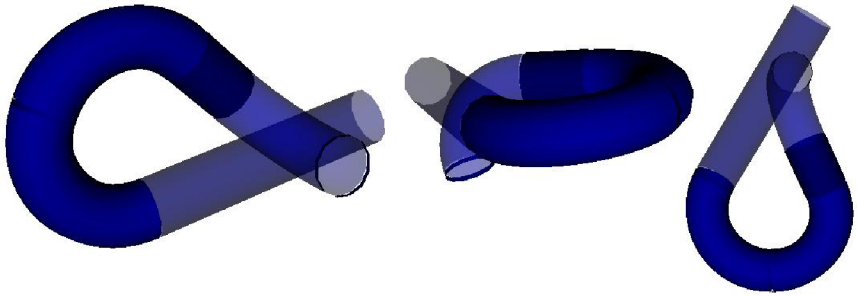
- **Počáteční napojení na hlavu motoru**
  Osa trubky musí vycházet ze středu hlavy válce $\mathbf{A}$ pod vektorem $\mathbf{m}$. Body $\mathbf{P}_1^1$ a $\mathbf{V}^1$ musí ležet na přímce definované bodem $\mathbf{A}$ a vektorem $\mathbf{m}$. Opět můžeme ověřit vyřešením soustav rovnic:

$$\mathbf{P}_1^1 = \mathbf{A} + s\cdot\mathbf{m} \tag{3.10}$$
$$\mathbf{V}^1 = \mathbf{A} + t\cdot\mathbf{m} \tag{3.11}$$

  A pro vypočtené parametry s, t musí platit: $0 < s < t$

- **Koncové napojení na pozici kolektoru výfuku**
  Podobně, osa trubky musí končit ve středu pozice kolektoru $\mathbf{B}$ pod vektorem $\mathbf{n}$:

$$\mathbf{P}_2^c = \mathbf{B} + s\cdot\mathbf{n} \tag{3.12}$$
$$\mathbf{V}^c = \mathbf{B} + t\cdot\mathbf{n} \tag{3.13}$$
$$0 < s < t \tag{3.14}$$



Obrázek 3.3: Pohled na trubku ze tří různých stran. Kolidující úseky trubky jsou vyznačeny průsvitnou barvou.

Dalším problémem je možnost kolize částí jedné trubky. Při určitém tvaru trubky mohou některé části trubky narážet do jiných částí stejné trubky. Tato situace je znázorněna na obrázku 3.3. Detekce těchto kolizí může vycházet z výpočtu objemu průniků jednotlivých úseků nebo z výpočtu vzdáleností os jednotlivých úseků trubky. Zároveň musíme udržovat mezi částmi trubky minimální povolenou vzdálenost d_{min}. Způsob detekce a ohodnocení kolizí si představíme v návrhu řešení.

Kolize trubky vůči ostatním objektům U těchto omezení detekujeme případné kolize mezi trubkami navzájem a kolize s okolními díly či polohu mimo vymezeného prostoru. Kolizí trubek rozumíme situaci, kdy se trubky protínají nebo jejich vzdálenost je menší než d_{min}. Stejně jako kolize částí trubky i detekce a ohodnocení kolizí trubek navzájem bude předmětem návrhu řešení.

U prostorových omezení vyšetřujeme většinou průnik s okolními díly. Výpočet těchto omezujících podmínek je dán konkrétní reprezentací prostorových omezení a není předmětem této práce. Proto ji budeme nadále chápat jako externí funkci, která dokáže na základě geometrie trubky detekovat kolize či určit průnik s okolními díly. Jejím výsledkem je kladná hodnota, pokud trubka pohybuje vně vymezeného prostoru či koliduje s okolními objekty. V návrhu řešení si alespoň představíme tvar funkce ohodnocující porušení prostorových omezení. Při testování systému budeme muset tuto skutečnost zohlednit a vhodným způsobem modelovat jejich přítomnost.

3.4 Kritéria optimalizace

Na základě konzultace s odborníkem v oblasti návrhu výfukových svodů byly stanoveny tyto kriteriální funkce:

Minimální délka trubek Hodnota tohoto kritéria je rovna délce nejdelší trubky:

$$f_1(\mathbf{x}) = \max_{i=1..p}(\mathtt{len}(\boldsymbol{t}_i)) \tag{3.15}$$

Délku trubky $\mathtt{len}(\boldsymbol{t})$, kde $\boldsymbol{t}$ je posloupnost oblouků – reprezentace trubky zavedená v části kapitoly 3.2, spočítáme jakou součet délek úseček a kruhových oblouků na ose

trubky:

$$\texttt{len}(\boldsymbol{t}) = |\mathbf{A}\mathbf{P}_1^1| + \sum_{i=1}^{c-1}(|\mathbf{P}_2^i\mathbf{P}_1^{i+1}| + \varphi^i \cdot r) + \varphi^c \cdot r + |\mathbf{P}_2^c\mathbf{B})| \tag{3.16}$$

Trubky stejné délky Toto kritérium spočítáme jako rozdíl délky nejdelší a nejkratší trubky.

$$f_2(\mathbf{x}) = f_1(\mathbf{x}) - \min_{i=1..p}(len(\boldsymbol{t}_i)) \tag{3.17}$$

Minimální středové úhly Požadujeme aby součet středových úhlů oblouků byl co nejmenší:

$$f_3(\mathbf{x}) = \sum_{i=1}^{p}\sum_{j=1}^{c_i}\varphi_i^j \tag{3.18}$$

Z pohledu výfukových systémů se uvedená kritéria snaží usnadnit průchod plynů výfukovými svody. Kritéria považujeme za rovnocenná co se týče důležitosti. Výsledkem optimalizační metody proto musí být množina řešení s vysokou diverzitou v prostoru kritérií.

3.5 Úloha z pohledu optimalizace

Nyní si uvedeme předpokládané problematické části úlohy vycházející z analýzy kritérií a omezujících podmínek. Tyto kritická místa popíšeme v kontextu multikriteriální optimalizace:

- **Neznáme optimální množinu řešení.**
  Pro konkrétní příklad nemáme k dispozici množinu optimálních řešení. Neznáme ani Pareto frontu, tedy optimální množinu řešení v prostoru kritérií. To způsobí problém s měřením výkonnosti optimalizační metody a stanovením požadavků na ukončení optimalizační metody.

- **Složitý tvar oblasti (případně více oblastí) proveditelných řešení.**
  Z uvedených omezujících podmínek lze předpokládat, že oblasti přípustných řešení budou mít složitý tvar. Poměr přípustných řešení k prostoru všech řešení je výrazně menší než u těch nepřípustných, a to i v případě, kdybychom proměnné oboustranně omezili. Optimální nebo sub-optimální řešení se často nachází na hranicích této oblasti. Tato skutečnost klade vysoké nároky na začlenění omezujících podmínek do optimalizační metody.

- **Obtížnost úlohy záleží na vstupních parametrech.**
  Vstupní parametry úlohy ovlivňují zejména tvar oblasti přípustných řešení. Zatímco na jedné úloze může optimalizační metoda dosahovat uspokojivých výsledků, na jiné nikoli. Zároveň je nutné zvolit vhodné řídící parametry optimalizační metody pro každou úlohu.

- **Vysoký počet návrhových parametrů v závislosti na vstupních parametrech.**
  Typické úlohy z praxe obsahují desítky návrhových parametrů v závislosti na počtu trubek a počtu ohybů. Z hlediska použití optimalizační metody to klade důraz na vhodné zakódování řešení, tak aby počet prohledávaných parametrů byl co nejnižší a nevedl k porušení omezujících podmínek. Zároveň zvolená reprezentace musí

umožňovat proměnný počet návrhových parametrů (variabilní počet ohybů trubky). Přesto nám zůstane mnoho navzájem se ovlivňujících parametrů. Prohledávání v takovém prostoru parametrů je náročné. Je potřeba skloubit globální prohledávání, kdy se snažíme nalézt optimální řešení pro úlohu s mnoha minimy kriteriální funkce (multimodální úlohy), a lokální prohledávání, kdy se snažíme prohledat okolí slibného řešení.

3.6 Požadavky na implementaci

Vstup a výstup aplikace Vstupní parametry úlohy mohou být zadávány pomocí textového souboru. Poté aplikace na základě optimalizační metody vypočítá množinu Pareto optimálních řešení. Tuto množinu musí aplikace uložit. Uvnitř souboru s řešením bude uložena geometrie tvaru trubek a příslušná kritéria.

Časová náročnost Čas optimalizační metody by se měl pohybovat v řádu desítek minut, maximálně jednu hodinu. Smyslem optimalizace je nalézt suboptimální řešení v rozumném čase. Úspěšnost návrhu výfukových svodů totiž záleží i na vstupních parametrech jako poloměr ohybu, průměr trubky nebo pozice pro trubku na válci a kolektoru výfuku. Uživatel tedy bude s aplikací experimentovat.

Implementace MOEA Jelikož evoluční algoritmy patří mezi stochastické optimalizační algoritmy, jejich průběh se odvíjí od posloupnosti náhodných čísel. Musíme proto zajistit kvalitní generátor pseudonáhodných čísel. Dále by vnitřní implementace MOEA měla poskytovat vývojáři možnosti pro snadnou modifikaci a experimentování s různými variantami algoritmu. Řídící parametry evolučního algoritmu musí být modifikovatelné pomocí konfiguračního souboru.

Možnost krokovat optimalizační metodu Aplikace by měla umožnit přerušovat evoluční algoritmus po několika iteracích a zobrazovat populaci řešení a další informace pro potřeby experimentování a ladění.

Práce s různými jednotkami Čas výpočtu a výsledky optimalizace pro různé jednotky vstupu by se neměly příliš lišit. Připomeňme, že kvůli stochastickému generování řešení nedosáhneme stejného průběhu evoluce a tedy často ani stejných výsledků. Výstupní řešení by mělo být ve stejných jednotkách, v jakých byly definovány vstupní parametry.

Požadavky na přesnost výpočtů Přesnost výpočtů se týká zejména omezujících podmínek. Proveditelné řešení musí opravdu vyhovovat výrobním požadavkům na konstrukci trubek. Obecně musí platit, že řešení, které prohlásíme za proveditelné, musí být proveditelné i u nástrojů poskytujících přesnější výpočet.

Kapitola 4

Návrh řešení

Z provedené analýzy problému je jasné, že je potřeba vybrat metodu, která je schopna řešit komplexní problémy s omezujícími podmínkami a více kritérii. Dalším požadavkem je snaha nalézt globální množinu kompromisních řešení.

Multikriteriální evoluční algoritmy se ukázaly v řadě podobných aplikací jako vhodná metoda, jež nepotřebuje znát podrobnosti výpočtu kriteriálních funkcí či výpočtu omezujících podmínek. Tato metoda rovněž nabízí možnost rozšiřování či modifikace. Toho využijeme zejména při návrhu optimalizační metody. Budeme vycházet od jednoho ze zástupců MOEA, který vykazuje dobré výsledky na sadě obecných testovacích úloh (tzv. benchmarků). Avšak použití úplně stejného algoritmu by bylo problematické. Některé části algoritmu budou navrženy pro konkrétní úlohu optimalizace výfukových svodů, zejména způsob zakódování jedince a variační operátory. Tyto části podrobně popíšeme. Na základě tohoto návrhu experimentálně odhadneme vhodné řídící parametry algoritmu. Součástí této fáze bude rovněž vyhodnocení na testovací sadě úloh.

Při návrhu evolučního algoritmu pro optimalizaci výfukových svodů jsem vycházel z již existujícího algoritmu NSGA-II. Důvody, proč jsem zvolil právě tento MOEA, jsou následující:

- Byl několikrát úspěšně využit pro řešení praktických úloh v oblasti multikriteriální optimalizace. Také jsou k dispozici studie, které jej porovnávají s jinými MOEA na sadě testovacích funkcí.[1]

- Pořadí jedince v populaci je založeno na konceptu Pareto dominance, který nevyžaduje složitý návrh. V případě SOEA bychom museli jednotlivá kritéria agregovat do jediné fitness a výkonnost algoritmu by se odvíjela od správného nastavení vah.

- Disponuje jednoduchým mechanismem udržování diverzity aproximace Pareto fronty, který nevyžaduje žádný řídící parametr.

- Umožňuje snadnou modifikaci jednotlivých částí algoritmu, volbu zakódování a variačních operátorů. Tato vlastnost je velice důležitá pro zachycení všech podstatných aspektů optimalizace výfukových svodů a vybudování optimalizační metody „na míru“. Například vzhledem k proměnnému počtu ohybů trubky si zřejmě nevystačíme se základními typy chromozomů. Nad vhodným zakódováním je potom možné vytvořit specializované operátory vycházející z analytické geometrie.

[1] viz např. [21]

- Umožňuje jednoduché začlenění omezujících podmínek do definice dominance.
- Použití některých vysoce výkonných EA pro spojité problémy (např. CMA-ES[2]) je v řadě případů problematické. Důvodem je proměnný počet návrhových parametrů, neexistující varianta pro multikriteriální optimalizaci, neschopnost zpracovávat omezující podmínky nebo nemožnost modifikace.

V následujících sekcích představíme konkrétní podobu NSGA-II pro řešení optimalizace výfukových svodů: zakódování jedince, ohodnocení a začlenění omezujících podmínek, variační operátory a generování počáteční populace. Snažíme se o to, aby algoritmus obsahoval málo řídících parametrů, a pokud existují, měly by být deterministicky určeny na základě vstupních parametrů, případně dynamicky adaptované v průběhu evoluce. Cílem je co nejmenší citlivost výkonnosti optimalizace na vývojářem zadávané řídící parametry, které mohou vykazovat odlišnou míru výkonnosti u různých úloh.

4.1 Zakódování jedince

Při návrhu zakódování jedince se snažíme pokrýt některé omezující podmínky tak, aby při modifikaci genů nevznikali nepřípustní jedinci, případně aby šlo porušení omezujících podmínek jednoduše testovat.

Osu trubky budeme reprezentovat lomenou čárou tak, jak je znázorněno na obrázku 4.1. Krajními body lomené čáry jsou vstupní parametry $\mathbf{A}, \mathbf{B}$, zatímco vnitřní body $\mathbf{V}^1, ..., \mathbf{V}^c$ jsou součástí chromozómu. Ke každému z těchto vnitřních bodů $\mathbf{V}^i$ přísluší oblouk $\mathbf{a}^i$. Oblouky tedy umísťujeme mezi dvě sousedící úsečky. Tohle zjednodušení si můžeme dovolit, neboť pracujeme pouze s konvexními úhly menšími než 180°. Právě tento způsob popisu nám pomůže jednoduše zachytit tečnou návaznost oblouků.

Osa trubky musí vycházet z hlavy motoru pod vektorem $\mathbf{m}$ a končit v kolektoru výfuku pod vektorem $\mathbf{n}$. Můžeme proto omezit pohyblivost první a posledního z vnitřních bodů osy trubky, tj. $\mathbf{V}^1$ a $\mathbf{V}^c$:

$$\begin{aligned} \mathbf{V}^1(s) &= \mathbf{A} + s \cdot \mathbf{m} \\ \mathbf{V}^c(t) &= \mathbf{B} + t \cdot \mathbf{n} \\ s &> 0 \\ t &> 0 \end{aligned}$$

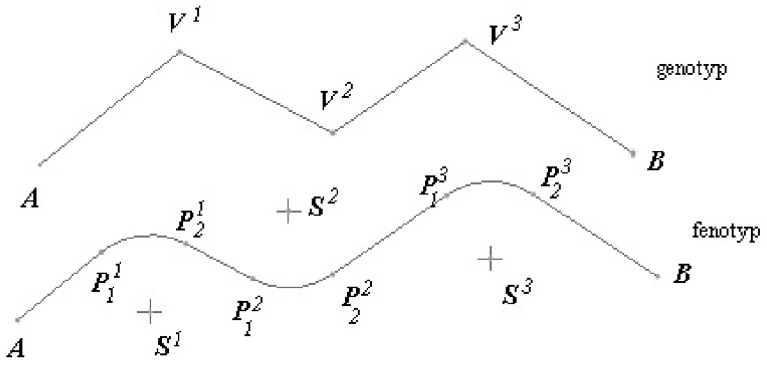
Vektory $\mathbf{m}, \mathbf{n}$ jsou zadávány v jednotkové velikosti, takže proměnná s určuje vzdálenost $\mathbf{V}^1$ od motoru a proměnná t určuje vzdálenost $\mathbf{V}^c$ od pozice kolektoru výfuku.

Jelikož středový úhel oblouku musí být větší než φ_{min}, můžeme stanovit minimální vzdálenost tečného bodu $\mathbf{P}$ a vrcholu $\mathbf{V}$ pro jakýkoli oblouk trubky. Tuto vzdálenost si označíme $|\mathbf{PV}|_{min}$. Její hodnotu vypočítáme pomocí vlastností pravoúhlého trojúhelníka:

$$|\mathbf{PV}|_{min} = r \cdot \tan\left(\frac{\varphi_{min}}{2}\right) \tag{4.1}$$

Nyní můžeme upravit výpočet bodů $\mathbf{V}^1, \mathbf{V}^c$ daný parametry s, t s použitím minimální vzdá-

[2]Covariance Matrix Adaptation Evolution Strategy



Obrázek 4.1: Znázornění zakódování trubky (genotypu) a jeho interpretace (fenotyp)

lenosti $|\mathbf{PV}|_{min}$:

$$\begin{aligned} \mathbf{V}^1(s) &= \mathbf{A} + (|\mathbf{PV}|_{min} + s) \cdot \mathbf{m} && (4.2) \\ \mathbf{V}^c(t) &= \mathbf{B} + (|\mathbf{PV}|_{min} + t) \cdot \mathbf{n} && (4.3) \\ s &> 0 \\ t &> 0 \end{aligned}$$

Genetickou informaci pro tyto dva body reprezentují parametry s, t. Pro ostatní vnitřní body lomené čáry je potřeba uchovat v genetické informaci všechny tři souřadnice. Chromozóm potom můžeme vyjádřit jako p vektorů reálných čísel o rozměrech $3 \cdot c_i - 4, i = 1..p$, kde c_i počet oblouků i-té trubky a p je počet trubek. Například pro trubku skládající se ze čtyř oblouků má vektor tento formát:

$$s, V_x^2, V_y^2, V_z^2, V_x^3, V_y^3, V_z^3, t$$

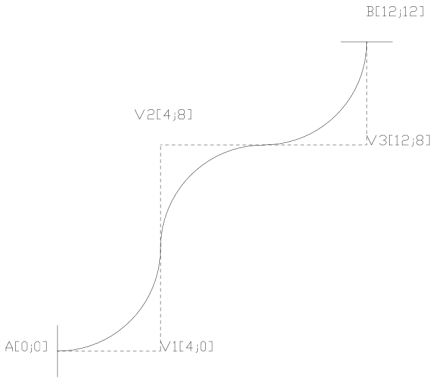
Celkem tedy chromozóm obsahuje $\sum_{i=0}^{p} 3 \cdot c_i - 4$ návrhových parametrů. Pokud bychom tuto formu chromozómu mapovat do jednorozměrného pole reálných čísel, je potřeba si pamatovat indexy, jež rozdělují jednotlivé definice trubek. Příklad v tabulce 4.1 demostruje zakódování osy trubky s třemi oblouky nacházejícími se ve stejné rovině a minimálním středovým úhlem oblouku $\varphi_{min} = \frac{\pi}{6}$.

Na základě této reprezentace – genotypu lze vypočítat fenotyp jedince, tj. řešení ve tvaru uvedeném v 3.2. Uvedeme si tedy výpočet oblouku (body $\mathbf{P}_1, \mathbf{P}_2, \mathbf{S}$, které tvoří definici oblouku zavedenou v 3.2) ze tří bodů lomené čáry $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$.

Výpočet středu oblouku je podobný výpočtu z 3.3:

$$\begin{aligned} \mathbf{u} &= \mathbf{V}_1 - \mathbf{V}_2 \\ \mathbf{v} &= \mathbf{V}_3 - \mathbf{V}_2 \\ \mathbf{w} &= \mathbf{u} + \mathbf{v} \\ \boldsymbol{S} &= \mathbf{V}_2 + t \cdot \mathbf{w} && (4.4) \end{aligned}$$

Hledáme parametr t, který určí hledaný bod $\mathbf{S}$, jež se nachází na ose oblouku. Využijeme znalosti goniometrických vzorců, vlastností pravoúhlého trojúhelníka a výpočtu odchylky



$$
\begin{aligned}
s &= |\mathbf{V^1A}| - \mathbf{PV}|_{min} \\
&= 4 - 4 \cdot \tan\left(\frac{\frac{\pi}{6}}{2}\right) \\
&= 2,9282
\end{aligned}
$$

chromozóm trubky:

$$\left[2,9282; 4; 8; 0; 2,9282\right]$$

Tabulka 4.1: Příklad zakódování trubky

dvou vektorů:

$$
\begin{aligned}
\cos(\phi) &= \frac{\mathbf{u} \cdot \mathbf{v}}{r^2} \\
\cos\left(\frac{\phi}{2}\right) &= \frac{|\mathbf{v}|}{t \cdot |\mathbf{w}|} \\
|\cos\left(\frac{\phi}{2}\right)| &= \sqrt{\frac{1 - \cos(\phi)}{2}}
\end{aligned}
$$

Po úpravě:

$$t = \frac{r}{|\mathbf{w}| \cdot \sqrt{\frac{1 - \frac{\mathbf{u} \cdot \mathbf{v}}{r^2}}{2}}} \tag{4.5}$$

Následuje výpočet tečného bodu $\mathbf{P}_1$, Hledáme průmět úsečky $\mathbf{V}_2\mathbf{S}$ na úsečku $\mathbf{V}_2\mathbf{V}_1$. Využijeme přitom geometrické interpretace skalárního součinu:

$$
\begin{aligned}
\mathbf{u} &= \mathbf{V}_1 - \mathbf{V}_2 \\
\mathbf{w} &= \mathbf{S} - \mathbf{V}_2 \\
\mathbf{P}_1 &= \mathbf{V}_2 + b \cdot \mathbf{u} \\
b &= \frac{|(\mathbf{V}_2 - \mathbf{P}_1(b))|}{|(\mathbf{V}_1 - \mathbf{V}_2)|} \\
b &= \frac{|\mathbf{w}| \cdot \cos(\frac{\pi - \varphi}{2})}{|\mathbf{u}|} \\
b &= \frac{\mathbf{w} \cdot \mathbf{u}}{\mathbf{u} \cdot \mathbf{u}}
\end{aligned} \tag{4.6}
$$

Druhý tečný bod $\mathbf{P}_2$ můžeme vypočítat podobně.

Problém zakódování oblouků s vysokým středovým úhlem Reprezentace trubky lomenou čárou umožňuje jednoduše vypočítat geometrickou definici trubky a detekovat případné porušení omezujících podmínek. Avšak čím je středový úhel oblouku větší, tím je bod

lomené čáry vzdálenější. Úhlů blízkých 180° nelze kvůli konečné reprezentaci čísel v počítačích dosáhnout. Vepsaný oblouk s vysokým středovým úhlem mezi úsečky lomené čáry se může ocitnout naprosto mimo vymezený prostor. Tato situace může způsobit chybu a tak je nutné se ji vyhnout. Přitom máme více možností jak to provést. Běžným řešením je analýza hranic, oblouků, které jsme ještě schopni zakódovat. My ale můžeme využít robustnosti evolučních algoritmů a tento problém vyřešit jednodušeji. Například tím, že odstraníme z populace řešení obsahujícího oblouky, jež nelze zakódovat, nebo se pokusíme znovu o vygenerování trubky či konkrétního oblouku. My při detekci oblouku, jež nelze reprezentovat bodem lomené čáry, ohodnotíme porušení omezujících podmínek pro příslušnou trubku nejvyšším možnou hodnotou.

4.2 Výpočet a začlenění omezujících podmínek

V části zabývající se analýzou omezujících podmínek (3.3) jsme si rozdělili omezující podmínky do dvou kategorií: omezení vycházející z konstrukce trubky a kolize trubky k okolním objektům. Na jemnější úrovni jsme rozdělili omezující podmínky takto:

- Tečná návaznost ohybů.
- Minimální úhel ohybu.
- Kolize částí trubky navzájem.
- Kolize trubek navzájem.
- Kolize trubek k okolním dílům.

Tohoto rozdělení využijeme i při návrhu dílčích funkcí ohodnocující porušení omezujících podmínek. V následujícím výkladu výpočtů omezujících podmínek budeme používat již zavedenou notaci z 3.2 a 3.3. Funkce a algoritmy budou pracovat jak se vstupními parametry (3.1), tak s genotypovými informacemi a fenotypem.

Pro ohodnocení porušení omezujících podmínek tečné návaznosti ohybů vycházíme ze zvolené reprezentace lomenou čárou. Potom nám stačí pouze kontrolovat, zda vzdálenost tečných bodů od vrcholu oblouku je menší než délka příslušného úseku lomené čáry. Pro každý oblouk tedy musíme vypočítat vzdálenost tečného bodu $|\mathbf{PV}|$ podle vzorce 4.6. Potom pro každý úsek lomené čáry kromě prvního a posledního vypočítáme rozdíl délky úseku a součtu vzdáleností $|\mathbf{PV}|$ pro oblouky, které tento úsek spojuje. Pro první a poslední úsek lomené čáry přísluší pouze jeden ohyb. Výsledné rozdíly sečteme. Funkci pro jednu trubku $\texttt{bindings}(\boldsymbol{t})$ proto můžeme formulovat takto:

$$\begin{aligned}\texttt{bindings}(\boldsymbol{t}) \;=\; & \max(|\mathbf{P}_1^1\mathbf{V}^1| - |\mathbf{A}\mathbf{V}^1|, 0) + \\ & + \sum_{i=1}^{c-1} \max(|\mathbf{V}^{i+1}\mathbf{P}_1^{i+1}| + |\mathbf{V}^i\mathbf{P}_2^i)| - |\mathbf{V}_{i+1}\mathbf{V}_i|, 0) + \\ & + \max(|\mathbf{P}_2^c\mathbf{V}^c| - |\mathbf{V}^c\mathbf{B}|, 0) \end{aligned} \tag{4.7}$$

Ohodnocení celého řešení potom vypočítáme součtem ohodnocení pro jednotlivé trubky:

$$\texttt{bindings}(\mathbf{x}) = \sum_{i=1}^{p} \texttt{bindings}(\boldsymbol{t}_i) \tag{4.8}$$

Všechny ohyby na sebe tečně navazují, právě když $\texttt{bindings}(\mathbf{x}) = 0$.

Výpočet ohodnocení porušení velikosti středových úhlů je navržen tak, abychom se vyhnuli výpočtu středového úhlu oblouku. Jelikož máme k dispozici minimální vzdálenost tečného bodu a vrcholu oblouku $|\mathbf{PV}|_{min}$, rozdíl vzdálenosti tečného bodu a vrcholu a $|\mathbf{PV}|_{min}$ nám potom udává míru porušení tohoto omezení:

$$\texttt{angles}(\boldsymbol{t}) = \sum_{i=0}^{c} \max(|\mathbf{PV}|_{min} - |\mathbf{P}_1^i \mathbf{V}^i|, 0) \tag{4.9}$$

Ohodnocení celého řešení potom vypočítáme součtem ohodnocení pro jednotlivé trubky:

$$\texttt{angles}(\mathbf{x}) = \sum_{i=1}^{p} \texttt{angles}(\boldsymbol{t}_i) \tag{4.10}$$

Kolize částí trubky navzájem je další překážkou konstrukční proveditelnosti trubky. Potřebujeme navrhnout funkci, která tuto omezující podmínku nejen testuje ale i ohodnotí její porušení. Navržený algoritmus pro vyhodnocení kolizí v trubce vyšetřuje vzdálenosti částí trubky. Trubka se skládá s rovných a obloukových úseků. Mezi dvěma obloukovými úseky se vždy nachází rovný úsek (může být nulové délky). Na začátku a na konci trubky je také rovný úsek. Celkově má tedy trubka $2 \cdot c + 1$, kde c je počet ohybů trubky. Budeme vyšetřovat kolize mezi úseky trubky, přitom často odmyslíme skutečnost, zda je úsek rovný nebo obloukový. Kolize dvou úseků počítáme na základě vzdálenosti jejich os (osy válce, obloukové osy). Je-li tato vzdálenost menší než $2 \cdot r + d_{min}$, části nedodržují minimální povolenou vzdálenost nebo se dokonce protínají.

U testování dvou úseků stejné trubky se ale situace komplikuje. Například vzdálenost dvou sousedících úseků je nulová, přesto mezi nimi žádné kolize nejsou. Proto testujeme pouze některé dvojice úseků, u kterých součet středových úhlů ohybů mezi nimi je větší než 180°. Vyšetřované úseky trubky musí na sebe tečně navazovat. Výpočet ohodnocení kolizí částí trubky je popsán v algoritmu 5. Pro každý úsek nalezneme takový oblouk, že součet středových úhlů mezi úsekem a obloukem včetně oblouku je větší nebo roven 180°. Pokud je větší, tak oblouk rozdělíme a vyšetřujeme pouze vzdálenost osy úseku a pravé části oblouku. Rozdělení oblouku provede operace $\texttt{split-arc}(\psi, \mathbf{a})$, která vrátí bod na oblouku $\mathbf{a}$ po jeho rozdělení na základě úhlu ψ. $(\mathbf{P}_1, \mathbf{P}_2, \mathbf{S})$ je konstruktor konvexního oblouku se středem $\mathbf{S}$ a tečnými body $\mathbf{P}_1, \mathbf{P}_2$. Operace $\texttt{dist-arc-part}(\mathbf{a}, i)$ vypočítá minimální vzdálenost os oblouku $\mathbf{a}$ a i-té části trubky. Poté testujeme úseky následující po tomto oblouku. Operace $\texttt{dist-parts}(i, j)$ počítá minimální vzdálenost os i-tého a j-tého úseku. Celkové ohodnocení kolizí v trubce je navrženo jako maximum přes všechny přírůstky porušení minimální vzdálenosti os vyšetřovaných úseků.

Tento navržený algoritmus je zde uveden v jednoduché podobě. Avšak může být dále optimalizován tak, aby nedocházelo ke zbytečným výpočtům. Například pokud součet úhlů mezi prvním a posledním úsekem je menší než 180°, nemusíme již žádné další úseky vyšetřovat. Je např. možné testovat dvojice na základě předpočítané „sumy prefixů“ úhlů a tím zrychlit výpočet.

Výpočtům pro vzdálenost os dvou úseků (vzdálenost dvou úseček, vzdálenost dvou oblouků, vzdálenost úsečky a oblouku) se budeme věnovat v části kapitoly zabývající se implementaci optimalizační metody.

Algoritmus 5 Ohodnocení kolizí trubky `pipe-parts-intersections`($\boldsymbol{t}$)

Vstup: $\boldsymbol{t}$ Vyšetřovaná trubka splňující podmínku tečné návaznosti ohybů.

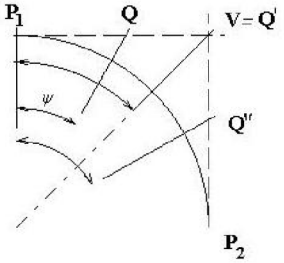
Výstup: *violation* Ohodnocení kolizí částí trubky. Pokud je rovno 0, trubka neobsahuje kolize.

```
violation ← 0
for i = 1 to 2 · c + 1 do
    sumAngles ← 0 // pro každý úsek trubky
    j ← ⌈(i+1)/2⌉
    while j < c ∧ sumAngles < π do
        sumAngles ← sumAngles + φ_j // Najdi první oblouk..
        j ← j + 1
    end while
    if sumAngles ≥ π then
        testedPart ← 2 · j − 1
        if sumAngles > π then
            Q ← split-arc(sumAngles − π, a_{j−i}) // Potřeba rozdělit oblouk
            violation ← max(violation, 2 · r + d_min − dist-arc-part((Q, P_2^{j−1}, S^{j−1}), i))
        end if
        for testedPart to 2 · c + 1 do
            violation ← max(violation, 2 · r + d_min − dist-parts(testedPart, i))
        end for
    end if
end for
```

Celkové ohodnocení kolizí částí trubky je pro jedince $\mathbf{x}$ dáno součtem ohodnocení pro jednotlivé trubky:

$$\texttt{pipe-parts-intersections}(\mathbf{x}) = \sum_{i=1}^{p} \texttt{pipe-parts-intersections}(\boldsymbol{t}_i) \tag{4.11}$$

Podle následujících vzorců lze určit bod $\mathbf{P}$, jež leží na oblouku a rozděluje jej podle úhlu ψ tak, že ramena $\mathbf{SP}_1$ a $\mathbf{SP}$ svírají úhel ψ a ramena $\mathbf{SP}$ a $\mathbf{SP}_2$ svírají úhel $\varphi - \psi$, kde φ je středový úhel oblouku. Princip výpočtu je znázorněn na obrázku.



$$\begin{aligned}
\mathbf{u} &= \mathbf{P}_1 - \mathbf{V} \\
\mathbf{v} &= \mathbf{P}_2 - \mathbf{V} \\
d &= |\mathbf{u}| = |\mathbf{v}| \\
\mathbf{Q} &= \begin{cases} \psi < \frac{\varphi}{2}, & \mathbf{V} + \mathbf{u} \cdot \left(\frac{d - r \cdot \tan(\psi)}{d}\right) \\ \psi = \frac{\varphi}{2}, & \mathbf{V} \\ \psi > \frac{\varphi}{2}, & \mathbf{V} + \mathbf{v} \cdot \left(\frac{d - r \cdot \tan(\varphi - \psi)}{d}\right) \end{cases} \\
\mathbf{w} &= \mathbf{Q} - \mathbf{S} \\
\mathbf{P} &= \mathbf{S} + \mathbf{w} \cdot \left(\frac{r}{|\mathbf{w}|}\right)
\end{aligned} \tag{4.12}$$

Na základě pozice trubky v prostoru musíme řešit její případné kolize s ostatními trubkami. Opět pro ohodnocení kolizí trubek využijeme výpočtu vzdálenosti os dvou úseků. Vyšetřované trubky musí vyhovovat tečné návaznosti ohybů.

$$\texttt{pipes-collisions}(\mathbf{x}) = \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \max(\max_{k=1..c_i, l=1..c_j} (2 \cdot r + d_{min} - \texttt{dist-parts}(i,k,j,l)), 0) \tag{4.13}$$

Funkce $\texttt{dist-parts}(i,k,j,l)$ vrací vzdálenost os k-tého úseku i-té trubky a l-tého úseku j-té trubky. Vybereme maximum přírůstků porušení kolizí mezi úseky trubek. Poté sečteme hodnoty pro všech $\binom{p}{2}$ kombinací trubek.

Výpočet kolizí vzhledem k okolním dílům není předmětem této práce, avšak nastíníme alespoň tvar funkce ohodnocující kolize k objektům či prostorovým omezením představených v sadě testovacích úloh, jež mají modelovat díly vyskytující se v okolí výfukových svodů.

Máme množinu prostorových omezení $o_1, .., o_n$, kde n je počet těchto omezení. Potom funkce $\texttt{pipe-obj-collisions}(\boldsymbol{t}, o)$ vrací nulu v případě, že trubka $\boldsymbol{t}$ neobsahuje žádné kolize s objektem (omezením) o. Pokud obsahuje kolize, funkce vrací kladnou hodnotu vyjadřující míru kolize. Celkové ohodnocení jedince $\mathbf{x}$ je potom dáno součtem dílčích ohodnocení přes všechny trubky a omezení:

$$\texttt{spatial-collisions}(\mathbf{x}) = \sum_{i=1}^{p} \sum_{j=1}^{n} \texttt{pipe-obj-collisions}(\boldsymbol{t}_i, o_i) \tag{4.14}$$

$$\texttt{pipe-obj-collisions}(\boldsymbol{t}, o) = \begin{cases} 0, & \boldsymbol{t} \cap o = \varnothing \\ > 0, & \boldsymbol{t} \cap o \neq \varnothing \end{cases} \tag{4.15}$$

Ohodnocení porušení omezujících podmínek je potřeba nyní začlenit do optimalizační metody, konkrétně do evolučního algoritmu NSGA-II. Tento krok je velmi důležitý, ovlivňuje konvergenci k žádané množině řešení. Toto začlenění je navrženo takto:

- Vygenerujeme počáteční populaci jedinců, kteří se skládají z trubek, jež splňují podmínku tečné návaznosti ohybů a podmínku velikosti středových úhlů, tzn. $\texttt{bindings}(\mathbf{x}) = 0 \wedge \texttt{angles}(\mathbf{x}) = 0$. Generování populace se podrobně věnujeme v samostatné části kapitoly (4.4).

- V iteracích evolučního algoritmu pracujeme již pouze s trubkami s tečně navazujícími ohyby a korektními středovými úhly. Požadujeme proto po variačních operátorech, aby generovaly pouze takovéto jedince.

- Kolize částí trubky, kolize trubek navzájem a kolize trubek s okolními díly budou sloučeny do jedné omezující podmínky $g(\mathbf{x})$, která tak bude vyjadřovat celkovou míru porušení omezujících podmínek jedince $\mathbf{x}$:

$$g(\mathbf{x}) = \texttt{pipe-parts-intersections}(\mathbf{x}) + \texttt{pipes-collisions}(\mathbf{x}) + \texttt{spatial-collisions}(\mathbf{x}) \tag{4.16}$$

  Tuto míru porušení omezujících podmínek poté zavedeme do definice dominance tak, jak jsme uvedli v kapitole popisující NSGA-II.

Návrh tohoto začlenění byl vytvořen na základě povahy jednotlivých omezení. K výpočtu veškerých kolizí je potřeba, aby trubky splňovaly tečnou návaznost ohybů. Přesto lze očekávat, že jedinců nesplňujících tuto podmínku je mnohem více než těch druhých. Zařazení veškerých kolizí do NSGA-II předpokládá prvotní fázi algoritmu, kdy několik iterací bude algoritmus hledat přípustného jedince, tzn. takového, který neobsahuje žádné kolize. Generovat takového jedince již v počáteční populaci by bylo náročné. Pokud algoritmus najde jedince bez kolizí, následuje prohledávání v prostoru kritérií.

4.3 Variační operátory

Účelem variačních operátorů je generovat nové jedince. Obvykle pracují s chromozómy rodičovských jedinců. Algoritmus 6 popisuje princip generování nové populace P' z rodičovské populace P. Křížení dvou jedinců provádíme s pravděpodobnosti p_{cross}. S pravděpodobností $1 - p_{cross}$ zkopírujeme do populace potomků původní jedince. Poté provedeme mutaci každého jedince.

Algoritmus 6 Vytvoření nové populace $P' =$ `new-pop`(P)

Vstup: $P = p_1, p_2, ...$ Rodičovská populace.
Výstup: P' Populace potomků.

1: **for** $i = 1$**to** $\frac{|P|}{2}$ **do**
2: **if** $\mathcal{U}(0,1) < p_{cross}$ **then**
3: $P' \leftarrow P' \cup$ `crossover`(p_{2i-1}, p_{2i})
4: **else**
5: $P' \leftarrow P' \cup p_{2i-1} \cup p_{2i}$
6: **end if**
7: **end for**
8: **for all** $q \in P'$ **do**
9: `mutation`(q)
10: **end for**

O vlivu křížení a mutace v evolučních algoritmech panují mezi odborníky rozdílné názory. Evoluční programování používá ke generaci nového jedince pouze mutaci a křížení považuje za nesmysluplné rozbíjení stavebních bloku. Jedince pokládá za jedinečnou kombinaci genů, kterou nelze křížit s nějakým jiným jedincem. Namísto toho používá propracovanější operátor mutace, který je často založen na normálním rozložení pravděpodobnosti a odchylky tohoto rozložení jsou obvykle součástí chromozomu.

Jiným extrémem jsou klasické genetické algoritmy, které považují křížení za hlavní nástroj pro generování jedinců s vysokým potenciálem uspět v další generaci. Tento přístup je dokonce formálně potvrzen teorémem o schématech. Mutace se používá pouze k zajištění diverzity populace jako prevence proti uváznutí v lokálním minimu. Tento přístup ovšem vykazuje špatné výsledky u tzv. klamných problému, kdy optimální řešení je složeno z podprůměrně ohodnocených schémat.

Návrh variačních operátorů pro úlohu optimalizace výfukových svodů se přiklání více k principům evoluční strategie a evolučního programování. Hlavním nástrojem pro vytvoření nového jedince bude tedy operátor mutace. Křížení bude mít pouze doplňující úlohu jako jediná výměna genů mezi jedinci v populaci.

Připomeňme požadavek na variační operátory z hlediska omezujících podmínek: nově vygenerovaný jedinec být složen z trubek splňujících tečnou návaznost ohybů a podmínku

velikosti středových úhlů.

4.3.1 Křížení

Návrh operátoru křížení spočívá v rekombinaci chromozómu po jednotlivých trubkách. Jednotlivé body křížení budou rozdělovat části chromozómu pro jednotlivé trubky. Noví jedinci poté vzniknou uniformní kombinací trubek rodičovských jedinců. Výhodou tohoto přístupu spočívá v jednoduché implementaci a faktu, že nový jedinec vyhovuje stanovenému požadavku na omezující podmínky. Pro každou trubku se náhodně vybere z rodičů jedinec, jehož trubka se zkopíruje do nového jedince. Můžeme takto vytvořit dva jedince s různou kombinací trubek. Avšak použití tohoto operátoru bez mutace by neposkytovalo dostatek nových jedinců.

4.3.2 Mutace

Algoritmus 7 Mutace jedince `mutation`($\mathbf{x}_{chrom}$)

Vstup: $\mathbf{x}_{chrom}$ Chromozóm jedince $\mathbf{x}$ ve tvaru, jak jsme jej představili v 4.1.
Výstup: $\mathbf{x}'_{chrom}$ Chromozóm nově vygenerovaného jedince.

1: **for** $i = 1$ **to** p **do**
2: $nTry \leftarrow 0$
3: **repeat**
4: $\boldsymbol{t}'_i \leftarrow$ `sample-pipe`($\mathbf{x}_{chrom}.\boldsymbol{t}_i$)
5: $nTry \leftarrow nTry + 1$
6: **until** $\neg$`is-feasible`($\boldsymbol{t}'_i$) $\wedge\ nTry < TMAX$
7: **if** $nTry = TMAX$ **then**
8: `Chyba: Nelze vygenerovat trubku.`
9: **else**
10: $\mathbf{x}'_{chrom}.\boldsymbol{t}_i \leftarrow \boldsymbol{t}'_i$
11: **end if**
12: **end for**

Algoritmus 7 popisuje základní princip navrženého operátoru mutace. Každou trubku generujeme do té doby, než se nám podaří vygenerovat trubku splňující požadované omezující podmínky, což testujeme pomocí funkce `is-feasible`($\boldsymbol{t}$) : `bindings`($\boldsymbol{t}$) $= 0 \wedge$ `angles`($\boldsymbol{t}$) $= 0$. K tomu máme pouze určitý maximální počet pokusů $TMAX$, jinak dojde k chybě. Připomeňme, že v mutaci pracujeme pouze s chromozómem jedince, takže pro kontrolu omezujících podmínek je potřeba provést převod na fenotyp.

Funkce `sample-pipe`($\boldsymbol{t}_{chrom}$) reprezentuje modifikaci trubky na základě jejího chromozómu. Funkce je popsána algoritmem 8. Skládá se celkem ze tří částí: odebrání oblouku, přidání oblouku a modifikace genů náhodnou hodnotou dle normálního rozložení pravděpodobnosti.

Nejprve s pravděpodobností p_{remove} provede odebrání náhodně vybraného oblouku (funkce `remove-arc`($\mathbf{a}$)). Nikdy neodebíráme první resp. poslední oblouk, protože bychom tím ztratili napojení na motor resp. kolektor. Vyjmout tedy můžeme pouze vnitřní oblouk tím, že z chromozómu odebereme všechny tři souřadnice vrcholu příslušného oblouku.

Poté s pravděpodobností p_{halve} provedeme rozdělení náhodně vybraného oblouku (funkce `halve-arc`($\mathbf{a}$)). Pro rozdělení oblouku nejdříve vypočítáme geometrii oblouku na základě

Algoritmus 8 Mutace trubky `sample-pipe`($\boldsymbol{t}_{chrom}$)

Vstup: $\boldsymbol{t}_{chrom}$ Část chromozómu pro příslušnou trubku.
Výstup: $\boldsymbol{t}'_{chrom}$ Chromozóm nově vygenerované trubky.

```
1: t'_chrom ← t_chrom
2: if c > c_min ∧ U(0,1) < p_remove then
3:    j = U[2, t'_chrom.c − 1]
4:    remove-arc(t'_chrom.a_j) // Vymaž jeden z vnitřních ohybů
5: end if
6: if c < c_max ∧ U(0,1) < p_halve then
7:    j = U[1, t'_chrom.c]
8:    halve-arc(t'_chrom.a_j) // Rozděl jeden z ohybů
9: end if
10: for i = 1 to 3 · t'.c − 4 do
11:    if U(0,1) < p_mut then
12:       g_i ← N(g_i, σ_mut) // Mutace genu normálním rozložením
13:    end if
14: end for
```

vrcholu $\mathbf{V}$ (viz výpočet fenotypu jedince v předchozí části kapitoly). Poté vypočítáme nové vrcholy $\mathbf{V}_1, \mathbf{V}_2$.

$$
\begin{aligned}
a &= |\mathbf{SV}| - r \\
b &= \frac{|\mathbf{P}_1\mathbf{V}|^2 - a^2}{2 \cdot |\mathbf{P}_1\mathbf{V}|} \\
\mathbf{u} &= \mathbf{V} - \mathbf{P}_1 \\
\mathbf{v} &= \mathbf{V} - \mathbf{P}_2 \\
\mathbf{V}_1 &= \mathbf{P}_1 + \mathbf{u} \cdot \frac{b}{|\mathbf{u}|} \qquad (4.17) \\
\mathbf{V}_2 &= \mathbf{P}_2 + \mathbf{v} \cdot \frac{b}{|\mathbf{v}|} \qquad (4.18)
\end{aligned}
$$

V zakódování trubky potom nahradíme předchozí vrchol $\mathbf{V}$ dvěma novými vrcholy $\mathbf{V}_1, \mathbf{V}_2$. Poznamenejme, že tato část mutace nezmění geometrii trubky, pouze její reprezentaci. Je však východiskem pro další části mutace a má významnou úlohu v prohledávání nových řešení. Následující příklad popisuje chromozóm trubky z tabulky 4.1 po rozdělení 2. oblouku:

$$\left[2,9282;\underline{4;8;0};2,9282\right] \rightarrow \left[2,9282;\underline{4;5,2928;0};\underline{6,7071;8;0};2,9282\right]$$

Nakonec s pravděpodobností p_{mut} provedeme mutaci genu pomocí náhodné hodnoty. Parametr σ_{mut} je odchylka normálního rozložení pravděpodobnosti. Z definice genotypu již víme, že geny jsou vlastně souřadnice bodů lomené čáry reprezentující trubku[3].

Z popisu navržené mutace je zřejmé, že jednotlivé části mutace mohou interferovat. Pravděpodobnost takových případů je dána součinem dílčích pravděpodobností.

Podíváme-li se na dílčí části mutace z hlediska porušení omezujících podmínek, rozdělení oblouku může narušit pouze podmínku velikosti úhlů, tečná návaznost zůstává zachována.

[3]Platí kromě prvního resp. posledního oblouku, kdy je v genotypu uchována vzdálenost k motoru resp. kolektoru.

Zatímco odebrání oblouku či mutace normálním rozložením může způsobit porušení obou podmínek.

Návrh mutace vychází z konkrétní povahy úlohy a několika experimentů s modelem výfukových svodů. Zejména od částí mutace, které mění počet oblouků trubky, si slibujeme vliv na diverzitu množiny řešení, což je důležité pro dosažení globální aproximace Pareto fronty. Mutace souřadnic bodů lomené čáry provádí drobné změny, lokálně prohledává prostor slibných řešení.

4.4 Generování počáteční populace

Pro inicializaci NSGA-II je potřeba vygenerovat počáteční populaci jedinců. Tato populace by měla být různorodá, abychom pokryli co největší prostor možných řešení. Dalším důležitým požadavkem, který jsme si stanovili, je splnění podmínky tečné návaznosti oblouků trubek a podmínky velikosti úhlů.

4.4.1 Generování trubky

Na začátku potřebujeme nějakým způsobem vytvořit lomenou čáru, která spojuje pozici na motoru s pozicí na kolektoru výfuku. Jelikož známe vektor, pod kterým má trubka vycházet z motoru resp. kolektoru výfuku, začneme právě generováním krajních oblouků. To provedeme pomocí generátoru spojitého rovnoměrného rozložení $\mathcal{U}(min, max)$. Zároveň náhodně zvolíme počet oblouků trubky pomocí generátoru diskrétního rovnoměrného rozložení $\hat{\mathcal{U}}(min, max)$.

$$s = \mathcal{U}(0, \delta) \tag{4.19}$$
$$t = \mathcal{U}(0, \delta) \tag{4.20}$$
$$c = \hat{\mathcal{U}}(c_{min}, c_{max}) \tag{4.21}$$

Převedeme-li geny s, t na fenotyp, vznikne nám trubka skládající se ze dvou oblouků. Mezi tyto dva oblouky vložíme další oblouky tak, abychom dosáhli požadovaného počtu oblouků. Vrcholy $\mathbf{V}^1, \mathbf{V}^c$ tedy označují body lomené čáry reprezentující první a poslední oblouk. Ostatní body rovnoměrně rozložíme mezi body $\mathbf{V}^1, \mathbf{V}^c$:

$$\mathbf{u} = \mathbf{V}^c - \mathbf{V}^1$$
$$\mathbf{V}^i = \mathbf{V}^i + \mathbf{u} \cdot \frac{i}{c-1} \quad i = 2, ..., c-1 \tag{4.22}$$
$$\tag{4.23}$$

Body poté posuneme o náhodně vygenerovaný vektor podle rovnoměrného rozložení.

$$\mathbf{V}^i \leftarrow \mathbf{V}^i + \mathcal{U}^{\mathbf{i}}(0, \sigma_{noise}) \quad i = 2, ..., c-1 \tag{4.24}$$

Takto náhodně vygenerujeme genotyp reprezentující trubku. Diverzitu generovaných trubek je možné ovlivnit pomocí parametrů δ a σ_{noise}.

4.4.2 Evoluce počáteční populace pro NSGA-II

Uvedený způsob generování trubky nám samozřejmě nezajistí splnění podmínky tečné návaznosti oblouků ani podmínku velikosti úhlů. Opakované generování trubky pro získání trubky

splňující obě dvě podmínky je možné řešení jak docílit žádaných trubek. Experimentálně jsme ale zjistili, že je k tomu potřeba velkého množství pokusů a u některých testovacích úloh nedosáhneme řešení v rozumném čase. Proto jsme se rozhodli pro vygenerování takového jedince použít jednoduchý optimalizátor opět s využitím EA. Použijeme jednokriteriální RCGA[4], který narozdíl od klasického GA používá zakódování pomocí reálných čísel. Toto označení chápeme spíše obecněji, protože jak uvidíme, jeho návrh je podobný principům evoluční strategie či evolučnímu programování. Typ chromozómu a jeho převod na fenotyp je stejný jako v případě NSGA-II. Představíme tedy stavební prvky algoritmu, vycházíme z kostry evolučního algoritmu (viz 2.2):

- Kriteriální funkce $f(\mathbf{x})$ pro ohodnocení jedince $\mathbf{x}$ je dána součtem funkcí vyhodnocujících porušení omezujících podmínek:

$$f(\mathbf{x}) = \texttt{bindings}(\mathbf{x}) + \texttt{angles}(\mathbf{x}) \tag{4.25}$$

- Počáteční populaci vytvoříme tak, že každou trubku vygenerujeme způsobem uvedeným v předchozí části (4.4.1) . Velikost populace se bude pohybovat v jednotkách maximálně desítkách jedinců.

- Jedince určené pro generování nových potomků vybíráme náhodně podle rovnoměrného rozložení. Přitom některý jedinec může být vybrán víckrát. Velikost rodičovské populace bude dvojnásobná než velikost populace. Vysoký selekční tlak tak zajistí rychlou konvergenci optimalizátoru.

- Z variačních operátorů použijeme pouze mutaci. Návrh operátoru mutace je přitom stejný jako u NSGA-II s tím rozdílem, že výsledné trubky nemusí splňovat žádnou z omezujících podmínek.

- Pro obnovu populace byl zvolen elitistický přístup. Aktuální populace je sloučena s populací potomků. Poté je vzestupně seřazena podle kriteriální funkce a ořezaná na původní velikost.

Pro získání každého jedince počáteční populace NSGA-II provedeme jeden běh RCGA. Tím dosáhneme nejvyšší diverzity počáteční populace.

[4]Real Coded Genetic Algorithm

Kapitola 5

Implementace a testování systému

V této kapitole popíšeme implementaci navržené optimalizační metody. Dále se budeme zabývat jejím testováním. Provedeme výběr sady testovacích vstupních dat. Představíme způsob měření výkonnosti multikriteriálních evolučních algoritmů. Podstatnou částí kapitoly je také nastavení řídících parametrů metody. Nakonec zhodnotíme dosažené výsledky na vybrané sadě testovacích úloh.

5.1 Implementace systému

Navrženou optimalizační metodu jsme implementovali v C++ za použití vývojového prostředí Visual Studio 2005.

Obecné analytické výpočty jsou implementovány v knihovně `GeomLib` a jsou tak odděleny od implementace evolučního algoritmu a výpočtů spojených s výfukovými svody, jež jsou obsaženy v knihovně `EvoAlg`.

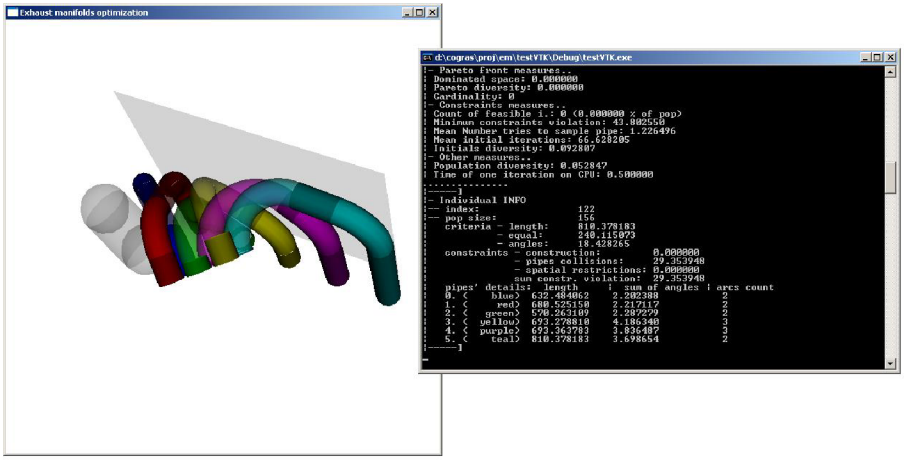
Načítání řídících parametrů z konfiguračního souboru, inicializace optimalizační metody a vizualizace včetně interakce s uživatelem jsou implementovány programem `emopt`. Vizualizace výfukových svodů je implementována pomocí knihovny VTK (viz `www.vtk.org`). Vstupní parametry úlohy včetně prostorových omezení jsou načteny pomocí textového souboru.

Systém dokáže pracovat s různými jednotkami, řídící parametry jsou v případě potřeby převedeny na jednotky, v nichž jsou definovány vstupní parametry.

Aplikace poskytuje uživateli dva režimy spuštění:

- Režim pro krokování evolučního algoritmu. Na začátku je vygenerována a zobrazena počáteční populace NSGA-II. Poté uživatel nechá provést určitý počet kroků evolučního algoritmu. Po této sérií iterací program zobrazí míru kvality aktuální populace. Uživatel taktéž může prohlížet tuto populaci nebo zobrazit chromozóm určitého jedince.

- Režim pro použití aplikace v praxi. Tento režim vyžaduje, abychom nalezli co možná nejlepší množinu řešení. Abychom zajistili co nejlepší globální prohledání prostoru možných řešení, můžeme provést více simulačních běhů algoritmu. Jedná se o nejjednodušší techniku, jak zajistit nalezení optimálních řešení. Na konci jsou aproximace Pareto množin z jednotlivých simulačních běhů sloučeny a vytvořena množina nedominovaných řešení. Poté je možné prohlížet tuto množinu, případně uložit některé řešení do souboru. Na konci simulačních běhů jsou vyhodnoceny míry kvality aproximace



Obrázek 5.1: Ukázka vytvořené aplikace v režimu krokování evolučního algoritmu

Pareto množiny a uloženy včetně dosažené Pareto fronty do textových souborů. To je důležité zejména pro pozdější vyhodnocení výsledků a výpočet statistických veličin.

Podrobný manuál pro ovládání aplikace je k dispozici na přiloženém CD. Aplikace byla otestována na operačním systému Windows XP.

5.1.1 Implementace analytických výpočtů

Výpočty z analytické geometrie jsou koncentrovány v jmenném prostoru Geometry. Třída `cgrMath` zatřešuje elementární matematické operace a konstanty. Základní geometrické entity jako vektor, bod či úsečka jsou reprezentovány třídami `Vect3D`, `Point3D` a `Segment3D`. Třída `CircleArc` implementuje reprezentaci kruhového oblouku a operace s ním spojené.

Protože úspěšnost evolučních algoritmů jako stochastické metody prohledávání do velké míry závisí na kvalitě posloupnosti náhodných čísel, v implementaci byl použit již existující a dobře otestovaný generátor pseudonáhodných čísel *Mersenne twister*.

Pro detekce kolizí trubek jsou potřeba výpočty vzdáleností mezi osami trubek, jež se skládají z úseček a kruhových oblouků. Zatímco výpočet vzdálenosti dvou úseček lze vypočítat analyticky (algoritmus z `www.softsurfer.com`), vzdálenost k oblouku je nutné vypočítat numericky na základě určité přesnosti. Podrobně si vysvětlíme, jak jsme tento problém vyřešili za pomocí vzdálenosti dvou úseček.

Výpočet ohodnocení porušení prostorových omezení je implementován třídě `SpLimViolationCalc`. Ohodnocení se opírá o výpočet vzdálenosti osy trubky a daného objektu či roviny. V případě roviny je potřeba určit polohu úsečky vzhledem k rovině, případně určit její vzdálenost. Pro vyhodnocení polohy oblouku jako části osy trubky vzhledem k rovině se potom vychází z následujícího algoritmu, jež počítá vzdálenost mezi oblouky a vzdálenost oblouku a úsečky.

Výpočet vzdálenosti k oblouku Pro výpočet porušení omezujících podmínek není nutné počítat skutečnou vzdálenost dvou oblouků. Pro výpočet této vzdálenosti použijeme algoritmický přístup. Ten má za účel zjistit zda dva oblouky vyhovují určité minimální povolené vzdálenosti a pokud ji nevyhovují, zjistit tuto vzdálenost s ohledem na požado-

vanou přesnost. Hlavním cílem je tedy detekce kolize oblouků, nikoli přesné určení jejich vzdálenosti.

Při návrhu tohoto algoritmu jsme se inspirovali principem „Rozděl a panuj“, používaným např. při vykreslování Beziérových křivek algoritmem *de Casteljau*. Tento algoritmus dělí usečky řídícího polygonu křivky, výpočítá bod na křivce a rekurzivně pokračuje pro nově vzniklé křivky až do dosažení určité přesnosti. Při algoritmu pro detekci kolize dvou oblouků nebo oblouku a úsečky budeme postupovat podobným způsobem. Nejdříve si uvedeme obecný princip výpočtu, poté uvedeme pro jednoznačnou interpretaci pseudokódy algoritmů.

Připomeňme, že poloměr všech oblouků je pro osu trubky konstantní a je součtem poloměru průřezu trubky a poloměru ohybu: $r = r_c + r_b$.

Oblouk se snažíme aproximovat vhodným tělesem tak, abychom si ulehčili výpočet vzdálenosti. Zároveň je nutné stanovit chybu takovéto aproximace. Takovýmto tělesem je například válec s dvěma polokoulemi na koncích. Oblouk obalíme do válce a potom počítáme vzdálenost s tímto válcem. Potom vzdálenost k takovému objektu můžeme vypočítat velice snadno pomocí vzdálenosti k úsečce – ose válce. Od ní poté odečteme poloměr válce. Krajní body osy válce jsou krajními body oblouku $\mathbf{P}_1, \mathbf{P}_2$.

Určením poloměru válce můžeme označit maximální chybu, v případě, kdy vzdálenost k oblouku je menší než vzdálenost k ose válce. Tuto maximální chybu vypočteme podle Pythagorovy věty z krajních bodů oblouku a poloměru r:

$$\texttt{Err}(\mathbf{P}_1, \mathbf{P}_2) = r - \sqrt{r^2 - \frac{|\mathbf{P}_1\mathbf{P}_2|^2}{4}}$$

Může také dojít k situaci, kdy vzdálenost k oblouku je vyšší než vzdálenost k ose válce a v extremním případě se náš odhad (se započítanou chybou) může lišit až o $2 \cdot \texttt{Err}(\mathbf{P}_1, \mathbf{P}_2)$. Tato chyba přesnosti aproximace může způsobit, že trubka bude chybně považována za kolidující, a snažíme se ji proto shora omezit parametrem ε_{max}.

Rozdělením oblouku na menší oblouky dosáhneme menší chyby při vyšetřování vzdálenosti. Toto rozdělení nemusí být rovnoměrné ale může odrážet potřebu zpřesňovat výpočet v potenciálních místech nejbližšího kontaktu. To nám pomůže ke konstrukci rychlého algoritmu s adaptační přesností. Pokud zjistíme, že vzdálenost vyšetřovaných oblouků může být menší než určité povolené minimum thr, pak vypočteme vnitřní bod oblouku a podle něj rozdělíme oblouk na dva. Poté vyšetřujeme vzdálenost k nově vzniklým obloukům. Takto rekurzivně pokračujeme dál až do přesnosti ε_{max}. Pokud je oblouk dostatečně daleko, tj. vzdálenost válce je vyšší než thr, pak již nebudeme oblouk dále dělit.

Algoritmus 9 detekuje kolizi mezi dvěma oblouky. Celková maximální chyba detekce vzdálenosti se potom skládá z chyb aproximací vyšetřovaných oblouků. Algoritmus střídavě vybírá oblouk k rozdělení na menší část. Pořadí je dáno proměnnou *turn*. Algoritmus 10 detekující vzdálenost úsečky a oblouku vychází ze ze stejného principu.

Z pohledu časové náročnosti a přesnosti algoritmus vykazuje dobré experimentální výsledky pro detekci kolizí mezi trubkami. Časovou náročnost jsem porovnal s přístupem, kdy oblouky byly teselovány na určitý počet bodů. Tento algoritmus počítá vzdálenosti mezi body oblouků a nejmenší z nich prohlásí za vyšetřovanou vzdálenost oblouků. Při stejné přesnosti byl nově navržený algoritmus pětkrát rychlejší. Často jsou totiž oblouky příliš vzdálené, a tak stačí pouze jeden výpočet vzdálenosti dvou úseček. Pokud bychom ale chtěli počítat přesnou vzdálenost mezi oblouky, algoritmus by nejspíše vykazoval vysokou časovou náročnost, protože by musel pokaždé vytvářet přesnou aproximaci oblouku.

Algoritmus 9 Vzdálenost dvou oblouků $\texttt{dist-arcs}(\mathbf{P}_1\mathbf{Q}_1\mathbf{S}_1, \mathbf{P}_2\mathbf{Q}_2\mathbf{S}_2, turn)$ menší než thr s maximální chybou ε_{max}

Vstup: $\mathbf{P}_1, \mathbf{Q}_1, \mathbf{S}_1, \mathbf{P}_2, \mathbf{Q}_2, \mathbf{S}_2 \in \mathbb{R}^3, turn = 1, 2$
Výstup: d

$\varepsilon_1 \leftarrow \texttt{Err}(\mathbf{P}_1, \mathbf{Q}_1)$
$\varepsilon_2 \leftarrow \texttt{Err}(\mathbf{P}_2, \mathbf{Q}_2)$
$s \leftarrow \texttt{dist-lines}(\mathbf{P}_1\mathbf{Q}_1, \mathbf{P}_2\mathbf{Q}_2)$
if $s \geq thr + \varepsilon_1 + \varepsilon_2$ **then**
    $d \leftarrow \infty$
else if $2 \cdot \varepsilon_1 < \varepsilon_{max} \wedge 2 \cdot \varepsilon_2 < \varepsilon_{max}$ **then**
    $d \leftarrow s - \varepsilon_1 - \varepsilon_2$
else
    if $2 \cdot \varepsilon_1 < \varepsilon_{max}$ **then**
        $i \leftarrow 2$
    else if $2 \cdot \varepsilon_2 < \varepsilon_{max}$ **then**
        $i \leftarrow 1$
    else
        $i \leftarrow turn$
    end if
    $\mathbf{v} = \frac{\mathbf{P}_i + \mathbf{Q}_i}{2} - \mathbf{S}_i$
    $\mathbf{M} \leftarrow \mathbf{S}_i + \mathbf{v} \cdot \frac{r}{|\mathbf{v}|}$
    $j \leftarrow i \mod 2 + 1$
    $d \leftarrow \min(\texttt{dist-arcs}(\mathbf{P}_i\mathbf{M}\mathbf{S}_i, \mathbf{P}_j\mathbf{Q}_j\mathbf{S}_j, j), \texttt{dist-arcs}(\mathbf{M}\mathbf{Q}_i\mathbf{S}_i, \mathbf{P}_jQ_j\mathbf{S}_j, j))$
end if

Algoritmus 10 Vzdálenost oblouku a úsečky $\texttt{dist-arc-seg}(\mathbf{PQS}, \mathbf{AB})$ menší než thr s maximální chybou ε_{max}

Vstup: $\mathbf{P}, \mathbf{Q}, \mathbf{S}, \mathbf{A}, \mathbf{B} \in \mathbb{R}^3$
Výstup: d

$\varepsilon \leftarrow \texttt{Err}(\mathbf{P}, \mathbf{Q})$
$s \leftarrow \texttt{dist-lines}(\mathbf{PQ}, \mathbf{AB})$
if $s \geq thr + \varepsilon$ **then**
    $d \leftarrow \infty$
else if $2 \cdot \varepsilon < \varepsilon_{max}$ **then**
    $d \leftarrow s - \varepsilon$
else
    $\mathbf{v} = \frac{\mathbf{P}+\mathbf{Q}}{2} - \mathbf{S}$
    $\mathbf{M} \leftarrow \mathbf{S}_i + \mathbf{v} \cdot \frac{r}{|\mathbf{v}|}$
    $d \leftarrow \min(\texttt{dist-arc-seg}(\mathbf{PMS}, \mathbf{AB}), \texttt{dist-arc-seg}(\mathbf{MQS}, \mathbf{AB}))$
end if

Co se týče detekce kolize mezi trubkami, algoritmus je navržen tak, aby nedošlo k porušení minimální vzdálenosti dvou trubek. Avšak může nastat situace, kdy vzdálenost trubek je větší než minimální vzdálenost, ale započítaná chyba aproximace způsobila, že jsou považovány za kolizní.

Oba představené algoritmy jsou implementovány ve třídě `DistanceArcAlgorithm`.

5.1.2 Implementace evolučního algoritmu

Implementace evolučního výpočtu byla provedena s ohledem na široké možnosti parametrizace EA. Z tohoto pohledu jsme často používali šablonového programování, protože jde o techniku, jak jednoduše měnit části optimalizační metody.

Šablona `TEA` představuje abstraktní třídu pro evoluční algoritmus a implementuje základní řízení algoritmu. Šablony `RCGA` a `NSGAII` implementují konkrétní algoritmy. Parametrizace těchto šablon umožňuje zvolit typ jedince a variační operátory. U `RCGA` je možno dále zvolit způsob výběru rodičovské populace a způsob obnovy. Obyčejná selekce náhodným výběrem je například implementována šablonou `UniformSelector`. Obnova populace ořezáním setříděné populace potomků s původními jedinci je implementována v šabloně `Elitism`.

Šablona `BaseIndividual` implementuje základní operace jedince: vytvoření, zánik, operace s chromozómem. Z ní jsou dále odvozeny třídy `EmVarLIndividual` a `FEmVarLIndividual` jež představují konkrétní typy jedinců. V prvním případě jde o jedince pro NSGA-II populaci, v druhém případě o jedince v RCGA populaci. Každý typ jedince může mít svoji formu chromozómu, musí implementovat ohodnocení řešení podle kritérií včetně omezujících podmínek a udržovat fenotyp, tedy konkrétní geometrické řešení. Šablona `TPipe` reprezentuje trubku, implementuje převod části chromozómu na trubku a vyhodnocuje omezující podmínky pro trubku. Toto vyhodnocení je uloženo ve struktuře `PenaltyFit`. Chromozóm je implementován třídou `EmVarLChromosome`. Ohodnocení řešení v prostoru kritérií je uloženo ve struktuře `EmFitness`.

Navržený operátor křížení jedinců je implementován v šabloně `EmVarLCrossover`, šablona `HybridMutator` implementuje operátor mutace. Tyto dva variační operátory pracují s typem chromozómu `EmVarLChromosome`.

Třída `InputParams` potom zapouzdřuje vstupní parametry úlohy, implementuje jejich načítání z textového souboru a převod do různých jednotek.

Míry pro měření výkonnosti MOEA představené v 5.3 jsou implementovány ve třídě `MOMeasures`. Pro výpočet míry S byl implementován algoritmus z [15].

5.2 Výběr testovací sady

V této části kapitoly představíme sadu úloh, pomocí níž budeme dále hodnotit vlastnosti optimalizační metody. Nejdříve formulujeme neformální požadavky na testovací úlohy:

- Testovací úlohy musí vycházet z praxe. Není sice možné otestovat všechny možné situace, důležité je ale otestovat stavy, které jsou pro návrh výfukových svodů typické.
- Úlohy by měly mít navzájem různou obtížnost – obsahovat jak běžné tak i některé extrémní případy, kde optimalizace nebude mít uspokojivé výsledky.
- Úlohy se musí od sebe lišit vstupními parametry – počtem trubek, vzájemnou polohou trubek, povoleným rozmezím počtu ohybů atd.

- Předpokladem vstupní úlohy je, že pozice začátku a konce trubky jsou vhodně zvolené. Výběr vhodného propojení hlav válců a pozic kolektorů výfuku totiž není předmětem optimalizace. To se týká i dalších vstupních parametrů jako poloměr ohybu nebo počet ohybů.
- Prostorová omezení musí vhodným způsobem modelovat přítomnost okolních dílů.

Tato práce se nezabývá extrakcí prostorových omezení z technického modelu a složitými vzdálenostními výpočty, jež jsou předmětem spíše integrace aplikace do CAD systému, ale budeme chápat tyto omezující podmínky obecněji a demonstrovat schopnost metody vypořádat se s tímto typem omezujících podmínek.

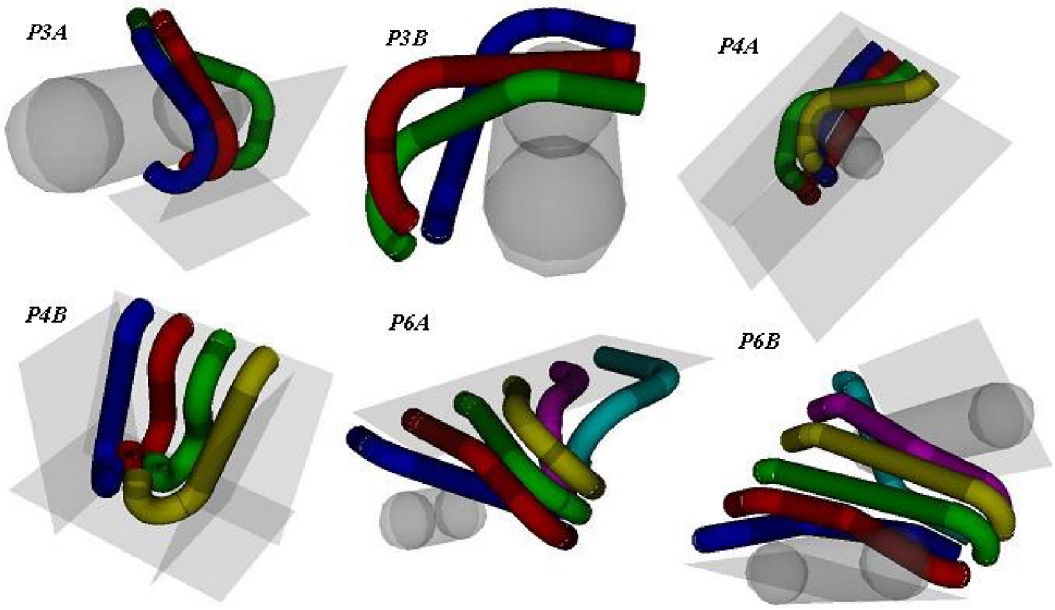
Prostorová omezení v testovací sadě proto modelujeme jednoduchými objekty jako koule, válec, případně omezeními dána rovinou. Cílem je jednoduše otestovat chování optimalizace návrhu trubek v omezeném prostoru, modelovat existenci okolních dílů v okolí výfukového systému. Ohodnocení porušení těchto omezujících podmínek je vypočítáno na základě minimální vzdálenosti osy trubky k rovině, středu koule, nebo k ose válce.

Zde jsou představeny hlavní testové úlohy:

Označení	**Počet trubek**	**Průměr trubky** [mm]	**Počet oblouků**	**Úhel ohybu** [°]	**Poloměr ohybu** [mm]	**Minimální vzdálenost trubek** [mm]
P3A	3	65	2-5	17-180	60	4
P3B	3	70	2-5	7-180	100	4
P4A	4	90	2-4	17-180	60	4
P4B	4	90	2-4	17-180	60	4
P6A	6	66	2-4	11-180	80	2
P6B	6	80	2-4	17-180	60	2

Testové úlohy se hlavně liší vzájemnou pozicí motoru a kolektoru výfuku a prostorovými omezení. Na obrázku 5.2 jsou znázorněna přípustná řešení pro jednotlivé testové úlohy včetně prostorových omezení. Následující tabulka neformálně hodnotí obtížnost jednotlivých úloh zejména z pohledu omezujících podmínek.

Úloha	**Komentář k omezujícím podmínkám**
P3A	Úloha klade nároky zejména na prvotní nalezení řešení splňujícího omezující konstrukce trubek, kdy součet úhlů pro trubku se často blíží 360°. Svody jsou omezeny rovinami ze dvou stran. Přítomnost motoru demonstruje válec.
P3B	Nenáročná prostorová omezení definována válcem, který modeluje přítomnost okolních dílů. Vyšší poloměr ohybu má zajistit menší délku trubek.
P4A	Prostorová omezení jsou daná rovinami ze tří stran a válcem. Úloha je náročnější pro splnění prostorových omezení a odstranění kolizí mezi trubkami.
P4B	Úloha je náročná na prvotní nalezení řešení splňujícího omezující podmínky konstrukce trubek. Prostorová omezení jsou dána rovinami ze čtyř stran.
P6A	Nenáročná prostorová omezení daná rovinou a válcem. Vzhledem k vyššímu počtu svodů řešení reprezentuje vysoký počet návrhových parametrů.
P6B	Jednoduchá úloha z hlediska generování trubek splňující omezující podmínky konstrukce. Prostorová omezení daná dvěma rovinami a dvěma válci mohou zkomplikovat nalezení množiny přípustných řešení a optimalizaci svodů podle kritérii.



Obrázek 5.2: Ukázka přípustných řešení pro jednotlivé testovací úlohy

Optimalizační metody budou testovány i na dalších úlohách vycházejících z představených šesti úloh. To nám pomůže lépe analyzovat chování optimalizační metody při změně vstupních parametrů a citlivost simulace optimalizačního algoritmu vzhledem k těmto parametrům. Výsledky experimentů budou ovšem uvedeny pouze na hlavních šesti ulohách.

5.3 Měření výkonnosti MOEA

Výkonnost optimalizačního algoritmu můžeme posuzovat ze dvou hledisek

- Výpočetní zdroje – čas.
- Kvalita výsledku.

Časovou náročnost EA můžeme měřit například počtem vyhodnocení fitness funkce nebo dobou běhu algoritmu na určitém stroji. Z tohoto hlediska není mezi SOEA a MOEA rozdíl.

Jiná situace nastává v případě měření kvality výsledku. Zatímco u SOEA často rozumíme kvalitou střední hodnotu fitness v populaci, u MOEA máme pouze koncept Pareto dominace, který dokáže porovnat pouze dva jedince, nikoli dvě množiny řešení. Není jasné, co se přesně myslí kvalitou aproximace Pareto fronty: Vzdálenost k optimálním řešením v prostoru kritérií, pokrytí širokého rozsahu různorodých řešení, nebo jiná míra. Proto je problematické definovat míry kvality aproximace Pareto fronty a důsledkem toho se často vykresluje získaná aproximace Pareto fronty do grafu.

Přesto tyto míry potřebujeme k tomu, bychom dokázali kvantitativně posoudit výsledky různých MOEA. Největší oblibu si získali unární míry kvality, tedy míry, které množině řešení přiřazují číslo na základě určitého hlediska. Příkladem je například velikost dominovaného objemu (S-míra), rozložení aproximace podél skutečné Pareto fronty nebo vzdálenost aproximace od skutečné Pareto fronty. Často je použito více měřítek a porovnávání dvou množin řešení je potom založeno na porovnávání měřítek pro tyto množiny.

Zitzler a spol. v [25] zkoumal míry kvality a prokázal teoretická omezení unárních mír. Zajímavé jsou zejména tyto výroky:

- Neexistuje žádná míra kvality, která je schopna potvrdit, zda aproximace Pareto fronty S je lepší než aproximace Pareto fronty T. Tento výrok platí i pro konečnou kombinaci unárních měřítek.
- Většina unárních mír, které byla navržena pro indikaci, že S je lepší než T, může nanejvýš indikovat, že S není horší než T, tj. S je lepší nebo nesrovnatelná vůči T.
- Binární míry kvality překonávají omezení unárních mír a pokud jsou vhodně navrženy, jsou schopné rozhodovat, zda S je lepší než T.

Dalším problémem je to, že neznáme skutečnou Pareto frontu, takže některé míry ani nemůžeme použít. Pro měření kvality aproximace Pareto fronty proto použijeme tyto tři často používané unární míry, jež budou podrobněji popsány v následujících sekcích:

- Míra S pro měření konvergence k Pareto frontě.
- Míra D pro měření diverzity uvnitř aproximace Pareto fronty.
- Kardinalita neboli počet řešení v aproximaci Pareto fronty.

Navíc pro porovnávání dvou množin řešení použijeme tyto dvě binární míry:

- Míra C (Coverage Relationship), jež k porovnávání využívá konceptu Pareto dominance.
- Míra D (Coverage Difference), jež je založena na S-míře.

Další míry budou představeny při vyhodnocení optimalizačního algoritmu na sadě testovacích úloh. V této práci jsme navíc vyvinuli míru F – *fenotypovou diverzitu*, která ohodnocuje množinu z hlediska různorodosti řešení výfukových svodů. Je založena na vzdálenosti dvou řešení. Její popis je uveden v následujících sekcích kapitoly.

Připomeňme, že evoluční algoritmy jsou stochastické prohledávací algoritmy vybavené generátorem náhodných čísel. Simulační běh algoritmu proto budeme muset několikrát opakovat a nad výsledky vypočítáme statistické veličiny jako střední hodnota a směrodatná odchylka.

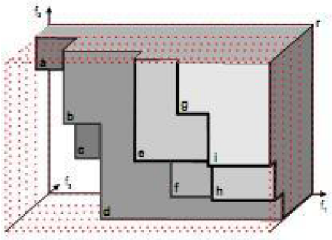
5.3.1 Míra S pro konvergenci

Metrika S byla poprvé představena v [24] a vyjadřuje objem prostoru dominovaného danou nedominovanou množinou řešení.

Definice 5.3.1 *Mějme pro uvažovaný problém množinu vektorů řešení* $\mathbf{X}$ *a* $\mathbf{A} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t \subseteq \mathbf{X}$ *množinu* t *vektorů řešení. Funkce* $S(\mathbf{A})$ *vrací objem uzavřený sjednocením polytopů* $p_1, p_2, ..., p_t$*, kde* p_i *je dán průnikem nadrovin vycházejících z* $\mathbf{x}_i$ *podél os: pro každou osu v prostoru kritérií existuje nadrovina kolmé k této ose a procházející bodem* $(f_1(\mathbf{x}_i), f_2(\mathbf{x}_i), ..., f_k(\mathbf{x}_i))$.

Pokud uvažujeme dvě množiny řešení, míra S je jediná míra, která ohodnotí lepší množinu vyšší hodnotou a vyšší hodnota znamená, že množina není horší.

Pro měření míry S je nutné zvolit referenční bod, jež je dominován všemi body aproximace Pareto fronty. Výpočet míry S není triviální, je záležitostí výpočetní geometrie. Existuje několik přístupů, které se liší zejména časovou náročností a omezením počtu kritérií [7], [2], [15].



Obrázek 5.3: Obrázek znázorňuje nedominovanou množinu devíti trojrozměrných bodů. Dominovaný objem je ohraničen referenčním bodem r.(Převzato z [2])

5.3.2 Míra E pro diverzitu aproximace Pareto fronty

Použijeme míru založenou na entropii [17]. Jedinci v populaci jsou podle podobnosti v prostoru kritérií rozděleni do několika tříd $C_1, C_2, ..., C_k$ tak, že $\forall i, j \in \{1, 2, ..., k\}, C_i \cap C_j = \varnothing \wedge \cup_{i=1}^{k} = P$. Odhad podobnosti vychází z euklidovské vzdálenosti vektorů v prostoru kritérií. Jednotlivá kritéria jedince normalizujeme podle minimální a maximální hodnoty kritéria v populaci:

$$f_i^{norm}(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{5.1}$$

Pokud pro dva jedince $\mathbf{x}$ a $\mathbf{x}'$ platí:$\|\mathbf{f^{norm}}(\mathbf{x}) - \mathbf{f^{norm}}(\mathbf{x}')\| < \sigma$, pak patří do stejné třídy. Po klasifikaci do tříd potom pro podmnožinu C_j musí platit:

$$\forall \mathbf{x} \in C_j \exists \mathbf{x}' \in C_j, \mathbf{x} \neq \mathbf{x}' : \|\mathbf{f^{norm}}(\mathbf{x}) - \mathbf{f^{norm}}(\mathbf{x}')\| < \sigma \tag{5.2}$$

Pro populaci obsahující N jedinců a optimalizačním problému o m kritériích se parametr σ počítá následovně:

$$\sigma = \sqrt{(1/N)^2 \cdot m} \tag{5.3}$$

Potom diverzitní míra E je definovaná takto:

$$E = \frac{-\sum_{j=1}^{k} P_j \cdot \log(P_j)}{E^{max}} \quad P_j = \frac{|C_j|}{N} \tag{5.4}$$

P_j je pravděpodobnost, že jedinec patří do třídy C_j. E^{max} je nejvyšší hodnota, která může nastat v čitateli, tj. v situaci, kdy $k = N$.

Cílem této míry je rovnoměrně rozložit jedince podél všech kritérií. Čím více druhů jedinců, tím větší hodnota míry E.

5.3.3 Míra F pro fenotypovou diverzitu

Míra F ohodnocuje různorodost řešení v populaci z hlediska vlastnosti výfukových svodů. Tato míra je založena na vzdálenosti dvou řešení. Připomeneme si formální požadavky na metriku $d(a, b)$:

- $d(a, b) \geq 0$
- $d(a, a) = 0$
- $d(a, b) = d(b, a)$

- $d(a,b) \leq d(a,h) + d(h,b)$

Existuje mnoho možností jak tuto metriku navrhnout. Obecným a často používaným způsobem je například euklidovská vzdálenost dvou vektorů reálných čísel reprezentujících řešení. Výpočet této vzdálenostní funkce je ale časové náročný kvůli vysoké dimenzi prostoru řešení. Navíc vektory mohou mít různý rozměr podle počtu oblouků trubky. Proto jsme se rozhodli postavit vzdálenostní funkci na ohodnocení odlišných vlastností trubky: délka trubky, součet středových úhlů ohybů trubky, počet ohybů. Ohodnocení těchto vlastností je navrženo následovně:

- Ohodnocení rozdílné délky dvou trubek $\boldsymbol{t}_1, \boldsymbol{t}_2$:

$$d_{length}(\boldsymbol{t}_1, \boldsymbol{t}_2) = 1 - \min\left(\frac{len(\boldsymbol{t}_1)}{len(\boldsymbol{t}_2)}, \frac{len(\boldsymbol{t}_2)}{len(\boldsymbol{t}_1)}\right) \tag{5.5}$$

  `len`($\boldsymbol{t}$) je délka osy trubky $\boldsymbol{t}$, výpočet je uveden v 3.16.

- Ohodnocení rozdílného součtu středových úhlů dvou trubek $\boldsymbol{t}_1, \boldsymbol{t}_2$:

$$d_{angles}(\boldsymbol{t}_1, \boldsymbol{t}_2) = \frac{|\sum_{i=1}^{c^{\boldsymbol{t}_1}}(\varphi_i^{\boldsymbol{t}_1}) - \sum_{i=1}^{c^{\boldsymbol{t}_2}}(\varphi_i^{\boldsymbol{t}_2})|}{\pi \cdot c_{max} - \varphi_{min} \cdot c_{min}} \tag{5.6}$$

- Ohodnocení rozdílného počtu ohybů dvou trubek $\boldsymbol{t}_1, \boldsymbol{t}_2$:

$$d_{bends}(\boldsymbol{t}_1, \boldsymbol{t}_2) = \frac{|c_{\boldsymbol{t}_1} - c_{\boldsymbol{t}_2}|}{c_{max} - c_{min}} \tag{5.7}$$

Každé z těchto dílčích ohodnocení je normalizované do intervalu $\langle 0, 1\rangle$. Vzdálenost dvou řešení $\mathbf{x}, \mathbf{y}$ potom určíme součtem dílčích ohodnocení trubek $\boldsymbol{t}_i^{\mathbf{x}}$ $\boldsymbol{t}_i^{\mathbf{y}}$ pro všech p trubek:

$$d(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{p}(d_{length}(\mathbf{x}.\boldsymbol{t}_i, \mathbf{y}.\boldsymbol{t}_i) + d_{angles}(\mathbf{x}.\boldsymbol{t}_i, \mathbf{y}.\boldsymbol{t}_i) + d_{bends}(\mathbf{x}.\boldsymbol{t}_i, \mathbf{y}.\boldsymbol{t}_i))}{3 \cdot p} \tag{5.8}$$

Míru F populace P potom vypočítáme jako sumu vzájemných vzdáleností mezi jedinci v P normalizovanou do intervalu $\langle 0, 1\rangle$:

$$F(P) = \frac{\sum_{i,j \in P \wedge i \neq j} d(i,j)}{\binom{n}{2}} \tag{5.9}$$

5.3.4 Míry C a D pro porovnání množin řešení

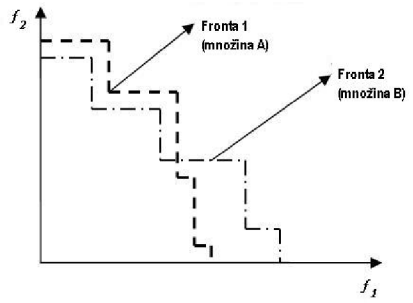
Míra C (Coverage of two sets) byla představena v [24] a slouží k porovnání dvou nedominovaných množin řešení.

Definice 5.3.2 *Mějme množinu vektorů řešení* $\mathbf{X}$ *pro uvažovaný problém a dvě množiny* $\mathbf{A}, \mathbf{B} \subseteq X$*. Funkce* C *mapuje uspořádaný pár* $(\mathbf{A}, \mathbf{B})$ *do intervalu* $\langle 0, 1\rangle$*:*

$$C(\mathbf{A}, \mathbf{B}) = \frac{|\mathbf{b} \in \mathbf{B} / \exists \mathbf{a} \in \mathbf{A} : \mathbf{a} \prec \mathbf{b}|}{|\mathbf{B}|} \tag{5.10}$$

Míru C potom interpretujeme následovně:

- $C(\mathbf{A}, \mathbf{B}) = 1$ znamená, že všechna řešení v $\mathbf{B}$ jsou dominování těmi z $\mathbf{A}$.



Obrázek 5.4: Graf dvou nedominovaných množin na dvou-kriteriálním problému.

- $C(\mathbf{A}, \mathbf{B}) = 0$ znamená, že žádné řešení v $\mathbf{B}$ není dominováno řešením z $\mathbf{A}$.
- $C(\mathbf{A}, \mathbf{B})$ nemusí být nutně rovno $C(\mathbf{B}, \mathbf{A})$.

Na obrázku 5.4 je znázorněna situace, kdy metrika C nemůže rozhodnout o kvalitě aproximace Pareto fronty. Proto byla navržena metrika D (Coverage difference of two sets), která uvažuje i rozdíl pokrytí prostoru dominovaných řešení.

Definice 5.3.3 *Mějme dvě nedominované množiny* $\mathbf{A}, \mathbf{B} \subseteq \mathbf{X}$. *Velikost prostoru dominovaného množinou* $\mathbf{A}$ *a nedominovaného množinou* $\mathbf{B}$ *je označován jako* $D(\mathbf{A}, \mathbf{B})$*:*

$$D(\mathbf{A}, \mathbf{B}) = S(\mathbf{A} + \mathbf{B}) - S(\mathbf{B}) \tag{5.11}$$

Pokud $D(\mathbf{A}, \mathbf{B}) > D(\mathbf{B}, \mathbf{A})$, tak fronta $\mathbf{A}$ dominuje frontu $\mathbf{B}$.

5.4 Odhad řídících parametrů

V této části kapitoly se pokusíme experimentálně odhadnout řídící parametry navržené optimalizační metody. K porovnávání jednotlivých nastavení použijeme jednak již představené míry kvality aproximace Pareto fronty, ale také další míry podle toho, jakou vlastnost algoritmu sledujeme.

5.4.1 Generování počáteční populace pomocí RCGA

Kvalitu RCGA budeme posuzovat na základě těchto mír:

- Fenotypová diverzita výsledné populace tvořena sloučením výsledků jednotlivých běhů RCGA. Chceme, aby tato hodnota byla co nejvyšší a tedy počáteční populace pro NSGA-II byla co nejvíce různorodá a pokrývala co nejvíce oblastí proveditelných jedinců.
- Dále chceme minimalizovat průměrný počet vyhodnocení omezujících podmínek pro dosažení hledaného jedince, což se projeví na časové náročnosti vygenerování počáteční populace. Můžeme jej odhadnout na základě velikosti populace potomků jako $2 \cdot N \cdot i$, kde i je průměrný počet iterací.

Budeme studovat vliv jednotlivých řídících parametrů RCGA na zmíněné míry. Poté odhadneme vztah mezi těmito parametry a vstupními parametry (počet trubek, počet oblouků, atd.).

Zavedeme si následující označení pro řídící parametry RCGA:

Název	**Popis**
N	Velikost populace RCGA
p_{mut}	Pravděpodobnost mutace genu
$p_{halvearc}$	Pravděpodobnost mutace trubky rozpůlením jednoho z oblouků.
$p_{removearc}$	Pravděpodobnost mutace trubky odebráním jednoho z oblouků.
σ_{mut}	Odchylka normálního rozložení pro mutaci genu.
σ_{noise}	Odchylka rovnoměrného rozložení pro mutaci souřadnice
δ	Maximální hodnota genu s, t pro generování prvního a posledního bodu lomené čáry reprezentující trubku

Velikost populace RCGA Experimentovali jsme s různými velikostmi populace RCGA pro různé testové úlohy. V tabulce 5.1 jsou výsledky experimentů pro úlohu P3A. V každém z případů bylo vytvořena počáteční populace o 40 jedincích. Z experimentů lze usoudit, že RCGA s menší populací musí provést více iterací, ale výsledná řešení od různých běhů jsou navzájem odlišná. Počet vyhodnocení je přitom nižší než u větší populace. Extrémní případ, kdy si algoritmus ukládá pouze jednoho jedince, vykazuje sice nejlepší výsledky, avšak často dochází k tomu, že řešení uvízne v lokálním minimu.

N	**Průměrný počet vyhodnocení**	**Diverzita F**
40	2400	0,1088
14	1022	0,1037
5	510	0,1142
3	408	0,1325
1	292	0,1383

Tabulka 5.1: Experimenty s různou velikostí populace na úloze P3A

Nastavení mutačního operátoru RCGA Navržený operátor mutace se skládá z několik částí, z nichž každá je provedena s určitou pravděpodobností. Jednotlivé části jsou: rozdělení oblouku trubky, odebrání oblouku, mutace souřadnic vnitřních bodů lomené čáry podle normálního rozložení. Nastavení mutačního operátoru se potom skládá z těchto řídících parametrů:

- **Pravděpodobnost rozdělení a odebrání ohybu** silně ovlivňuje schopnost mutace vytvářet odlišné jedince. Pokud by tyto dvě pravděpodobnosti byly nulové, počet oblouků by byl určen pouze na začátku náhodným vygenerováním. Bylo vypozorováno, že

pro některé úlohy se úspěšnost nalezení trubky splňující tečnou návaznost oblouků odvíjí od použitého počtu oblouků. Naopak vysoká hodnota pravděpodobnosti znamená příliš velké změny při mutaci trubky a RCGA poté pomalu nebo vůbec nekonverguje. Experimentováním bylo zjištěno, že tyto dvě pravděpodobnosti by měly být shodné, a měly by se pohybovat v intervalu 0,05 až 0,1. V tabulce 5.2 jsou uvedeny některé z experimentů pro úlohu P6A.

- **Pravděpodobnost mutace genu** trubky znamená pravděpodobnost modifikace souřadnice bodu lomené čáry normálním rozložením. Nastavení této pravděpodobnosti úzce souvisí s odchylkou normálního rozložení a ovlivňuje tak míru modifikace trubky.
- **Odchylka normálního rozložení pro mutaci genu** ovlivňuje jak rychlost konvergence RCGA tak odlišnost jednotlivých řešení. Malá odchylka znamená malou změnu trubky, algoritmus potom potřebuje více iterací k dosažení hledaného optima. Vyšší odchylka umožňuje globální prohledávání, diverzita populace uvnitř RCGA i diverzita výsledné počáteční populace je vyšší. Hodnota odchylky mezi 15 až 25 mm vykazovala nejlepší výsledky u obou mír.

$p_{halvearc} = p_{removearc}$	p_{mut}	**Průměrný počet vyhodnocení**	**Diverzita** F
0,1	1	474	0,1334
0,1	0,5	660	0,1291
0,1	0,7	576	0,1431
0,2	0,7	978	0,1329
0,05	0,7	462	0,1399
0,05	1	474	0,1334

Tabulka 5.2: Experimenty s různým nastavením mutace při stejné odchylce normálního rozložení pro úlohu P6A

Nastavení náhodného generování trubky zahrnuje parametry σ_{noise} a δ. Tyto parametry ovlivňují tvar trubek v počáteční populaci RCGA a jak bylo experimenty zjišteno, mají i výrazný vliv na tvar trubek v počáteční populaci v NSGA-II a následnou úspěšnost optimalizace. Pokusy o odhad řídícího parametru δ na základě vstupních parametru, např. definice začátku a konce trubky nebo prostorových omezení, negenerovaly trubky požadovaných tvarů. Proto je tento parametr určen konstantně pro všechny úlohy. Odchylka rovnoměrného rozložení σ_{noise} potom přidává do prvotního deterministického odhadu trubky určitý „šum“, čímž chceme generovat trubky různých tvarů. Příliš velká hodnota ale generuje dlouhé členité trubky.

V následující tabulce jsou experimentálně zjištěné hodnoty řídících parametrů. Většina řídících parametrů je konstantní pro všechny testové úlohy, pouze velikost populace RCGA je určena na základě počtu trubek p.

Název	Hodnota
N	$1 + \lfloor \frac{p}{2} \rfloor$
p_{mut}	0.8
$p_{halvearc}$	0.1
$p_{removearc}$	0.1
σ_{mut}	20 mm
σ_{noise}	10 mm
δ	100 mm

5.4.2 Multikriteriální optimalizace pomocí NSGA-II

Nyní budeme studovat vliv řídících parametrů NSGA-II na výslednou aproximaci Pareto fronty $\mathbf{P}^*$:

- Velikost dominovaného objemu $S(\mathbf{P}^*)$
- Diverzita $D(\mathbf{P}^*)$
- Kardinalita $|\mathbf{P}^*|$

Dále nás budou zajímat následující veličiny:

- Fenotypová diverzita (míra F) populace v průběhu evoluce.
- Průměrný počet vyhodnocení omezujících podmínek pro dosažení prvního jedince splňujícího všechny omezující podmínky.
- Průměrný počet pokusů pro vygenerování trubky splňující omezující podmínky konstrukce trubek v průběhu evoluce.

Řídící parametry si označíme podobně jako v předchozí části:

Název	**Popis**
N	Velikost populace
p_{cross}	Pravděpodobnost křížení dvou jedinců
p_{mut}	Pravděpodobnost mutace genu
$p_{halvearc}$	Pravděpodobnost mutace trubky rozpůlením jednoho z oblouků
$p_{removearc}$	Pravděpodobnost mutace trubky odebráním jednoho z oblouků
σ_{mut}	Odchylka normálního rozložení pro mutaci genu

Velikost populace NSGA-II Připomeňme, že pro NSGA-II velikost populace N určuje velikost rodičovské populace, na níž je aplikován operátor selekce, křížení a mutace. Populace potomků má tutéž velikost. Po sloučení rodičovské populace a populace potomků tedy dostáváme $2 \cdot N$ jedinců, z nichž je v další generaci znovu vytvořena rodičovská populace o velikosti N. Jelikož operátor křížení pracuje vždy po dvou jedincích, N musí být sudé.

Při experimentování s různou velikostí populace jsme algoritmus nechali běžet pro dostatečný počet generací (500), kdy už dlouho nedocházelo ke zlepšení kvality aproximace Pareto fronty. V tabulce 5.3 jsou výsledky některých experimentů pro úlohu P4A. Algoritmus s příliš malou velikostí populace hůře udržuje diverzitu uvnitř populace a tak často uvízne

v lokálním minimu. S příliš vysokou velikosti populace se výrazně zvyšuje počet vyhodnocení kritérií a tedy i časová náročnost algoritmu, zatímco kvalita aproximace Pareto fronty se nezvyšuje. Dále bylo zjištěno, že vhodná volba velikosti populace závisí na vstupních

N	$S(\mathbf{P}^*)[m^2 \cdot rad]$	$E(\mathbf{P}^*)$	$\lvert\mathbf{P}^*\rvert$
30	1.4986	0.911985	11.7
	0.2942	0.049808	3.8
60	1.9400	0.952986	21.5
	0.2767	0.032911	8.8
100	1.8241	0.903598	15.5
	0.1618	0.073757	8.3

Tabulka 5.3: Výsledky měření aproximace Pareto množiny při různé velikosti populace na úloze P4A

parametrech jako je počet trubek či počet možných oblouků u trubky. Následující vzorec pro určení vhodné velikosti populace je odhadnut na základě počtu trubek p a maximálního počtu ohybů i-té trubky c^i_{max}:

$$N = 2 \cdot \left\lceil 6 \cdot \ln(1 + 4 \cdot p) + \sum_{i=1}^{p} \ln(3 \cdot c^i_{max}) \right\rceil \tag{5.12}$$

Nastavení variačních operátorů NSGA-II Pravděpodobnost křížení dvou jedinců zejména ovlivňuje rychlost konvergence k hledanému optimu, proto u klasických GA je tato pravděpodobnost rovna jedné a křížení je tedy se základním operátorem pro generování nových řešení. Nevýhodou příliš velké rychlosti konvergence je ztráta diverzity populace a nebezpečí uváznutí v lokálním minimu. Toho se budeme snažit vyvarovat, zvláště při řešení více-kriteriálního problému, kdy výsledkem není pouze jedno řešení ale různorodá sada kompromisních řešení. Navržený operátor křížení produkuje nové jedince pouze kombinací trubek rodičovských jedinců, ale neprodukuje žádné nové tvary trubek. Proto mutační operátor bude hrát hlavní roli při generování populace potomků.

Přesto bylo experimenty s různou (i nulovou) pravděpodobností křížení zjištěno, že navržený operátor má pozitivní vliv na konvergenci algoritmu ke kvalitní aproximaci Pareto fronty. Pravděpodobnost křížení je nastavena konstantně:

$$p_{cross} = 0.8$$

Při experimentování s nastavením mutačního operátoru je potřeba zkoušet různé kombinace hodnot řídících parametrů, abychom identifikovali jejich vzájemné interakce.

Nastavení pro RCGA nedává dobré výsledky pro použití v NSGA-II. Příliš velká odchylka normálního rozložení a vysoká pravděpodobnost mutace brání prohledávání v okolí lokálního minima. Doporučené hodnoty pro parametr σ_{mut} se pohybují v rozmezí 5 až 12 mm a pro p_{mut} v rozmezí 0.2 až 0.5. Vyšší hodnota σ_{mut} pomáhá udržovat diverzitu v průběhu populace. Tato vlastnost je důležitá také pro prohledávání oblastí proveditelných řešení. Příliš vysoká hodnota p_{mut} ale brání konvergenci ke konečné množině řešení, kdy je potřeba jemnějšího prohledávání. Tomu se budeme snažit vyhnout tím, že budeme tento parametr v průběhu evoluce snižovat. Na začátku běhu algoritmu bude pravděpodobnost mutace nejvyšší, abychom co nejvíce pokryli prostor možných řešení. V pozdějších fázích evoluce bude

vliv mutace slábnout a dojde k lokálnímu prohledávání slibného řešení. Počáteční pravděpodobnost p_{mut} je experimentálně odvozena z maximální možné velikosti chromozómu. Pravděpodobnost v generaci g je potom dána funkcí $p_{decMut}(g)$.

$$p_{mut} = \frac{2}{\sqrt{\sum_{i=1}^{p} 3 \cdot c_{max}^{i} - 4}} \tag{5.13}$$

$$p_{decMut}(g) = \frac{p_{mut} \cdot (1 + a^{g})}{2} \quad a \sim 0.99 \tag{5.14}$$

Přitom standardní odchylka normálního rozložení pro mutaci souřadnice je nastavena konstantně:

$$\sigma_{mut} = 10mm$$

Parametry $p_{halvearc}$ a $p_{removearc}$ by měli mít stejnou hodnotu. Experimenty bylo zjištěno, že volbou různých hodnot $p_{halvearc}$ a $p_{removearc}$ můžeme ovlivnit počet ohybů u trubek určitým směrem. Odhad těchto parametrů na základě povoleného rozmezí počtu ohybů u trubek jsme zvolili následovně:

$$p_{halvearc} = p_{removearc} = \frac{1}{3 + \sum_{i=1}^{p} c_{max}^{p} - c_{min}^{p}} \tag{5.15}$$

5.5 Dosažené výsledky

Navržený evoluční algoritmus jsme otestovali na sadě šesti úloh. Abychom eliminovali stochastické efekty evolučního prohledávání, algoritmus jsme na každé testovací úloze spustili desetkrát. Z výsledků jednotlivých simulačních běhů jsme spočítali statistické veličiny: střední hodnotu a směrodatnou odchylku.

Směrodatná odchylka jako míra variability nám v tomto případě udává, jak moc se výsledky jednotlivých simulačních běhů od sebe liší a tedy jaký vliv má na evoluční algoritmus posloupnost náhodných čísel.

Zatímco velikost populace je odvozena na základě vstupních parametrů (zejména počtu trubek), počet iterací NSGA-II byl intuitivně zvolen na základě složitosti úlohy.

Řídící parametr	**P3A**	**P3B**	**P4A**	**P4B**	**P6A**	**P6B**
Velikost populace NSGA-II	54	54	62	62	78	78
Velikost populace RCGA	2	2	3	3	4	4
Počet iterací NSGA-II	400	400	500	500	500	500

Tabulka 5.4 udává míru kvality dosažené aproximace Pareto množiny. Pro ujasnění interpretace těchto výsledků zdůrazněme, že míru S mezi jednotlivými úlohami nelze porovnávat. Pro každou úlohu je Pareto fronta jiná a při výpočtu míry S je pro každou úlohu stanoven jiný referenční jedinec. Tyto výsledky nám ovšem pomohou porovnávat následné experimenty s tímto výchozím návrhem řešení, což je podstatným ukazatelem při rozhodování. Při interpretaci velikosti aproximace Pareto množiny musíme uvažovat nastavenou velikost populace.

Relativní míra $E(\mathbf{P}^{*})$ popisuje schopnost algoritmu generovat různorodou množinu řešení. Výsledky pro testovací úlohy ukazují, že se podařilo udržet střední hodnotu této míry nad 0.9. Avšak pro úlohu P4A je tato hodnota nejnižší stejně jako velikost aproximace Pareto množiny.

Dále vyhodnocujeme tyto vlastnosti algoritmu RCGA a NSGA-II:

Míra	P3A	P3B	P4A	P4B	P6A	P6B
$S(\mathbf{P}^*)[m^2 \cdot rad]$	2.647	3.161	2.239	2.322	0.1719	1.843
	0.04822	0.2376	0.219	0.02122	0.01003	0.1567
$E(\mathbf{P}^*)$	0.9499	0.9277	0.9024	0.9488	0.9158	0.9276
	0.02537	0.02414	0.07987	0.03434	0.0516	0.02582
$\lvert\mathbf{P}^*\rvert$	53.5	54.5	46.71	60.15	58.58	78.7
	4.188	0.6455	14.39	6.323	13.78	3.132

Tabulka 5.4: Výsledky měření aproximace Pareto pro testovací úlohy

Označení	Význam
$\mathcal{I}_{RCGA}$	Počet iterací RCGA pro dosažení jedince NSGA-II.
$F(P_{init})$	Fenotypová diverzita počáteční populace.
$\mathcal{I}_{NSGA}$	Počet iterací NSGA-II pro dosažení prvního přípustného řešení.
$F(P_{result})$	Fenotypová diverzita výsledné populace.
$\mathcal{T}$	Průměrný strojový čas běhu NSGA-II v sekundách.

Tabulka 5.5 potom obsahuje výsledky těchto veličin pro jednotlivé testovací úlohy. Z hlediska schopnosti algoritmu prohledávat v omezeném prostoru nás nejvíce zajímají míry $\mathcal{I}_{RCGA}$ a $\mathcal{I}_{NSGA}$. Nalezení trubek, jež vyhovují podmínkám konstrukce (tečná návaznost ohybů a velikost středových úhlů), nejdéle trvalo pro úlohu P4B – až 105 iterací RCGA. Zatímco nejrychlejší konvergence bylo dosaženo u příkladu P6B. Schopnost generovat počáteční populaci jedinců se tedy podstatně liší podle polohy pozic na motoru a na kolektoru výfuku.

Schopnost nalézt jedince NSGA-II, který vyhovuje všem omezujícím podmínkám (zejména těm prostorovým), se liší u jednotlivých testovacích úloh méně než $\mathcal{I}_{RCGA}$. Přesto například pro úlohu P4A docházelo k vysokým výkyvům v jednotlivých simulačních bězích.

Jednotlivé testovací úlohy se také výrazně liší časovou náročností (míra $\mathcal{T}$). To je pravděpodobně způsobeno různou velikostí chromozómu v závislosti na počtu trubek a obtížností omezujících podmínek.

Míra	P3A	P3B	P4A	P4B	P6A	P6B
$\mathcal{I}_{RCGA}$	68.47	38.27	43.34	105	69.22	14.94
	4.397	1.633	2.483	4.022	3.011	0.6443
$F(P_{init})$	0.1401	0.1204	0.1509	0.1467	0.1371	0.1564
	0.005745	0.006449	0.005317	0.003713	0.00304	0.002271
$\mathcal{I}_{NSGA}$	6.643	12.58	33.79	15.77	25.67	22.6
	5.259	2.812	24.54	5.727	7.063	8.499
$F(P_{result})$	0.04963	0.04917	0.02607	0.03073	0.04871	0.04728
	0.01448	0.01074	0.01468	0.01962	0.01469	0.008773
$\mathcal{T}$	50.58	29.66	71.02	105.4	105.5	135.9
	2.514	2.769	7.43	3.129	6.518	6.344

Tabulka 5.5: Výsledky měření aproximace Pareto pro testovací úlohy

Z pohledu návrhu výfukových svodů se nám podařilo pro všechny testovací úlohy nalézt řešení, které vyhovuje všem stanoveným omezujícím podmínkám. Na grafu dosažené apro-

ximace Pareto fronty pro příklad P6A (obrázek A.5) lze vidět vztahy jednotlivých kritérií. Každý z třech grafů znázorňuje vztah dvou kritérií z hlediska množiny nejlepších kompromisů. Hledání kompromisů protichůdných kritérií nejlépe ukazuje vrchní graf vlevo, kde výfukové svody s vyšším součtem středových úhlů ohybů mají trubky blízké délky. Na základě takového pozorování můžeme interpretovat vztahy mezi kritérii. K tomu, aby trubky měly stejnou délku je potřeba, aby trubka změnila tvar, což často způsobí zvýšení součtu středových úhlů. Naproti tomu spodní graf vlevo můžeme interpretovat tak, že zkrácením délky trubek často snížíme i součet středových úhlů a naopak. Tvar Pareto fronty může být pro každou testovací úlohu jiný, a proto tato interpretace nemusí platit. V příloze A jsou k dispozici dosažené aproximace pro všechny testovací úlohy.

Co se týče rozpětí jednotlivých kritérií, podařilo se ve většině případů udržet diverzitu řešení uvnitř aproximace Pareto množiny, což umožňuje posléze uživateli vybrat kompromisní řešení podle aktuálních preferencí. Požadavek na minimální rozpětí délky výfukových svodů (kritérium f_2) byl zřejmě nejvíce náročný. U úlohy P4A bylo dosaženo největšího rozpětí tohoto kritéria a aproximace Pareto množiny obsahuje u všech testovacích úloh kromě P6A řešení, které má trubky stejné délky.

Kapitola 6

Experimenty s nastavením evolučního algoritmu

Na základě dosažených výsledků při testování optimalizační metody jsme navrhli experimenty s návrhem algoritmu. Popis experimentu obsahuje hlavní myšlenku, motivaci pro provedení experimentu a předpokládané chování pozměněné metody. Zároveň uvedeme použité řídící parametry. Na konci kapitoly zhodnotíme dosažené výsledky na sadě testovacích úloh včetně přínosu nových variant algoritmu.

6.1 Experimenty se začleněním omezujících podmínek

Začlenění omezujících podmínek výrazně ovlivňuje schopnost algoritmu efektivně se pohybovat v omezeném prostoru. Tato kategorie experimentů se zabývá úpravou stávajícího začlenění omezení a analýzou chování algoritmu v těchto případech.

6.1.1 Tolerance velikosti středových úhlů

Snažíme se usnadnit přechody mezi oblastmi přípustných řešení. Přesuneme proto výpočet funkce ohodnocení porušení omezujících podmínek velikosti středových úhlů $\texttt{angles}(\mathbf{x})$ do definice dominance. V populaci NSGA-II se tak mohou vyskytovat jedinci nesplňující podmínku velikosti středových úhlů ohybu. Tato změna způsobí menší zatížení RCGA pro generování počáteční populace. V populaci NSGA-II bude ale větší množství nepřípustných řešení než v původním návrhu.

Původní výpočet porušení omezujících podmínek pro jedince v NSGA-II uvedený v upravíme následovně:

$$\begin{aligned} \mathtt{g}(\mathbf{x}) \quad = \quad & \texttt{angles}(\mathbf{x}) + \texttt{pipe-parts-intersections}(\mathbf{x}) + \\ & \texttt{pipes-collisions}(\mathbf{x}) + \texttt{spatial-collisions}(\mathbf{x}) \end{aligned} \tag{6.1}$$

Při tomto experimentu nebyly měněny žádné řídící parametry představené v 5.4.

6.1.2 Tolerance nedodrženého počtu ohybů

Z experimentů s mutačním operátorem plyne, že proměnlivost počtu ohybů trubky v průběhu evoluce je důležitým nástrojem prohledávání prostoru řešení. Při tomto experimentu budeme generovat trubky různého počtu oblouků i mimo interval $\langle c_{min}, c_{max} \rangle$. Přitom budeme jedince penalizovat určitou hodnotou, pokud obsahuje trubky, které mají počet ohybů

mimo povolený rozsah. Tuto pokutu zahrneme do funkce ohodnocující porušení omezujících podmínek. Takto se nám nemůže stát, že by výsledná množina řešení porušovala rozsahy počtu ohybů.

Pokuta za nedodržený počet ohybů na trubku $\boldsymbol{t}$ bude mít následující tvar

$$\texttt{bends-count}(\boldsymbol{t}) = q \cdot \max(c_{min} - c, c - c_{max}, 0)^2 \quad q \sim 40 \tag{6.2}$$

Pro jedince x je ohodnocení dáno součtem ohodnocení jeho trubek:

$$\texttt{bends-count}(\mathbf{x}) = \sum_{i=0}^{p} \texttt{bends-count}(\boldsymbol{t}_i) \tag{6.3}$$

Použitím exponenciální funkce se pokuta zvyšuje, pokud $a > 1$. Parametr q potom reprezentuje jakousi normalizační váhu vzhledem k ostatním částem ohodnocení porušení podmínek. Původní výpočet porušení omezení pro jedince v NSGA-II uvedený v totiž upravíme následovně:

$$\begin{aligned} \mathrm{g}(\mathbf{x}) \quad = \quad & \texttt{bends-count}(\mathbf{x}) + \texttt{pipe-parts-intersections}(\mathbf{x}) + \\ & \texttt{pipes-collisions}(\mathbf{x}) + \texttt{spatial-collisions}(\mathbf{x}) \end{aligned} \tag{6.4}$$

6.2 Experimenty s generováním nové populace

Tato kategorie experimentů se týká způsobu generování nových jedinců, tzn. změny ve výběru rodičovské populace, operátoru křížení nebo mutace.

6.2.1 Zvýšení populace potomků

NSGA-II má původně velikost populace stejnou jako velikost populace potomků. Následující experiment spočívá ve zvýšení selekčního tlaku na jedince tím, že potomků bude výrazně více než jedinců z původní populace. To můžeme provést hned několika způsoby:

- Úpravou variačních operátorů tak, aby produkovaly více jedinců. Například křížení by generovalo více jedinců v závislosti na počtu trubek. Jinou možností je úprava mutačního operátoru.
- Naplnění rodičovské populace více jedinci. Operátory tak zůstanou bez úpravy a počet potomků bude stejný jako velikost rodičovské populace.

Výběrem druhé varianty si značně usnadníme práci a efekt bude s největší pravděpodobností stejný. Budeme tedy zkoušet různé velikosti rodičovské populace v závislosti na velikosti

Každý jedinec má nyní větší šanci podílet se na genetické informaci budoucí generace. Také je pravděpodobné, že vlivem elitismu bude v příští generaci méně jedinců z předchozí generace než v původním návrhu NSGA-II. To na jedné straně zrychlí konvergenci, na druhé straně může dojít ke ztrátám slibných řešení v důsledku vysokého selekčního tlaku. Také musíme počítat s vyšší počtem vyhodnocení kriteriálních funkcí, což má za následek zpomalení evolučního algoritmu. Proto v experimentech snížíme počet iterací algoritmu tak, aby počet vyhodnocení byl stejný jako v původním návrhu. Potom je teprve možné porovnávat výsledky obou konfigurací algoritmu.

Experimentálně odvozený vztah mezi velikostí rodičovské populace M a velikostí populace N je dán následujícím vzorcem:

$$M = \left\lfloor \frac{3 \cdot N}{5} \right\rfloor \cdot 2 \tag{6.5}$$

Připomeňme, že velikost rodičovské populace vzhledem k binárnímu operátoru křížení musí být sudá.

6.2.2 Mutace s adaptačními odchylkami normálního rozložení

Úprava mutačního operátoru se týká odchylky normálního rozložení pro modifikaci souřadnice bodu lomené čáry. Pro návrh tohoto experimentu jsme se inspirovali u evoluční strategie a evolučního programování, které již ve své původní formě pracuje s adaptací odchylky. Tento řídící parametr σ_{mut} jsme v návrhu řešení určili konstantou. Nyní každý gen bude mít k dispozici svoji odchylku, která bude součástí chromozómu. Obecný chromozóm jedince potom má tvar $(x_1, x_2, ..., x_n, \sigma_1, \sigma_2, ..., \sigma_n)$, kde n je počet genů. Tyto odchylky budou měněny v průběhu evoluce při každé mutaci podle následujících vztahů:

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot \mathcal{N}(0,1) + \tau \cdot \mathcal{N}_i(0,1)) \tag{6.6}$$

$$\sigma_i' < \varepsilon \Rightarrow \sigma_i' = \varepsilon \tag{6.7}$$

Konstanta ε je minimální povolená odchylka. Parametry τ a τ' jsou konstanty ovlivňující rychlost učení. Čím vyšší hodnota těchto konstant, tím větší změny odchylek, tedy rychlejší učení. Při menších hodnotách se odchylky učí pomaleji. Doporučené hodnoty jsou určeny na základě počtu genů takto:

$$\tau' = \frac{2}{\sqrt{2 \cdot \sqrt{n}}} \tag{6.8}$$

$$\tau = \frac{2}{\sqrt{2 \cdot n}} \tag{6.9}$$

Potom jednotlivé geny jsou určeny následovně:

$$x_i' = x_i + \sigma_i \cdot \mathcal{N}(0,1) \tag{6.10}$$

Odchylky normálního rozložení v chromozómu jedince nesou tedy jakousi metainformaci o směru prohledávání v prostoru proměnných. Tento způsob generování jedinců je často zakomponován do evoluční strategie a má dobré výsledky u spojitých problémů. Základním předpokladem této strategie je to, že jedinci s dobrými odchylkami budou mít dobré ohodnocení a jejich genetická informace se tedy promítne do další generace. Uvažujeme-li tyto odchylky jako poloosy n-rozměrného elipsoidu, potom generování jedince představuje generování bodu uvnitř tohoto elipsoidu. Poloosy elipsoidu jsou přitom rovnoběžné s osami souřadnicového systému.

Z hlediska zakódování tvaru výfukových svodů jsou geny souřadnice bodů lomené čáry, jež reprezentuje trubku. Odchylky potom obsahují informaci o vhodných pozicích oblouku. Nesmíme ovšem zapomenout, že operátor mutace může rozdělit oblouk či ho naopak odebrat. Také připomeňme, že operátor mutace generuje řešení po jednotlivých trubkách a požaduje splnění omezujících podmínek tečné návaznosti oblouků a velikosti středových úhlů. Návrh adaptačních odchylek je proto nutné sladit s těmito skutečnostmi. Dále je potřeba odchylky inicializovat na určitou hodnotu σ_{init}.

Adaptace odchylek pouze na začátku celého procesu mutace jedince vykazovalo při experimentech problémy při generování trubek a často došlo k překročení maximálního počtu pokusů. Proto při neúspěšném pokusu o generování trubky znovu provedeme adaptaci odchylek z původního stavu. Počet pokusů se potom výrazně snížil.

Při rozdělení oblouku nám vzniknou nové body lomené čary, jež nenesou informaci o odchylkách souřadnic. Ty je nutné inicializovat na určitou hodnotu. Experimentálně bylo zjištěno, že inicializace na vysokou hodnotu způsobí velké změny v tvaru řešení a tím pravděpodobné vyloučení jedince z populace. Takto se ztrácí i genetická informace o rozdělení daného oblouku. Účinnost rozdělování oblouků v mutaci je tak nepřímo potlačována. Proto inicializujeme nové odchylky konstantou ε, tedy minimální povolenou odchylkou. Při odebrání oblouku neměníme odchylky souřadnic ostatních bodů.

Řídící parametry σ_{init} a ε jsou určeny konstantně pro všechny vstupy:

$$\begin{aligned} \sigma_{init} &= 10mm \\ \varepsilon &= 1mm \end{aligned}$$

6.2.3 Mutace na základě pořadí vrcholů

Chceme upravit mutační operátor tak, aby generoval trubku s tečně navazujícími ohyby a vyhovujícími středovými úhly již v operaci `sample-pipe` (algoritmus 8). Navržená změna mutace bude tvořit lomenou čáru postupně po bodech na základě určitého pořadí. Přitom kontroluje omezující podmínky a při neúspěchu umožňuje generovat bod znovu.

Pořadí vrcholů $\mathbf{o} = (o_1, o_2, ..., o_c)$ bude uloženo v chromozómu jako permutace, kde prvek permutace označuje index oblouku. Tato permutace je v mutaci modifikována na základě pravděpodobnosti $p_{arcsOrder}$ tak, že náhodně vybraný prvek permutace je prohozen se svým pravým sousedem.

Algoritmus 11 Generování lomené čáry `sample-vertex`$(\mathbf{o}, i)$

Vstup: $\mathbf{o}$ permutace pořadí vrcholů. i index v této permutaci – právě zpracovávaný oblouk.
Výstup: $TRUE$ pokud generování uspělo, jinak $FALSE$.

```
1: cnt ← 0
2: while ¬success ∧ cnt < ω do
3:    V^{o_i} ← modify-norm(V^{o_i})
4:    success ← check-constraints()
5:    cnt ← cnt + 1
6: end while
7: if success ∧ i < c then
8:    success ← sample-vertex(o, i + 1)
9: end if
10: return success
```

Algoritmus 11 popisuje princip generování bodů lomené čáry. Nové body lomené čáry jsou generovány v pořadí zakódovaném v chromozómu. Pokud nově vygenerovaný vrchol způsobí porušení omezujících podmínek (velikost středových úhlů nebo porušení tečné návaznosti ohybů) je proveden další pokus o jeho generování. Počet pokusů je ovšem ohraničen maximem ω. Poté se rekurzivně pokračuje v generování dalšího bodu podle určeného pořadí.

Operace `check-constraints`() zkontroluje proveditelnost trubky. Protože generujeme vrcholy oblouků postupně, nevíme jak bude trubka vypadat. Můžeme tedy kontrolovat pod-

mínky, máme-li k tomu dostatečný počet bodů. Představme si generování trubky o pěti ohybech. Čtyři body $\mathbf{V^1}, \mathbf{V^2}, \mathbf{V^4}, \mathbf{V^5}$ jsou již úspěšně generovány. Víme tedy, že ohyby u bodů $\mathbf{V^1}, \mathbf{V^5}$ mají proveditelné středové úhly. Také víme, že oblouk u bodu $\mathbf{V^2}$ tečně navazuje na oblouk u bodu $\mathbf{V^1}$, stejně jako u bodů $\mathbf{V^4}, \mathbf{V^5}$. Jelikož ještě nebyl generován bod $\mathbf{V^3}$, víc toho zkontrolovat nemůžeme. Avšak po jeho generování je nutné zkontrolovat středové úhly oblouků u bodů $\mathbf{V^2}, \mathbf{V^3}, \mathbf{V^4}$ a tečnou návaznost mezi oblouky u bodů $\mathbf{V^1}, \mathbf{V^2}, \mathbf{V^3}, \mathbf{V^4}, \mathbf{V^5}$, tj. celkem čtyři úsečky.

Funkce `modify-norm`($\mathbf{V}$) generuje nový bod. Vychází přitom z původního bodu $\mathbf{V}$. Generování bodu se provádí podle normálního rozložení pravděpodobnosti s odchylkou σ_{mut}. V případě prvního a posledního bodu lomené čáry generujeme pouze vzdálenost k motoru, resp. kolektoru výfuku:

$$\mathbf{V}^1 \leftarrow \mathbf{A} + \mathbf{m} \cdot (|\mathbf{PV}|_{min} + \sigma_{mut} \cdot \mathcal{N}(0,1)) \tag{6.11}$$

$$\mathbf{V}^c \leftarrow \mathbf{B} + \mathbf{n} \cdot (|\mathbf{PV}|_{min} + \sigma_{mut} \cdot \mathcal{N}(0,1)) \tag{6.12}$$

Pro ostatní body musíme generovat všechny tři souřadnice:

$$\mathbf{V}^i \leftarrow \mathbf{V}^i + \sigma_{mut} \cdot \mathcal{N}^{\mathbf{i}}(0,1) \tag{6.13}$$

Takto navržená mutace může v extrémním případě generovat až ω^c trubek. Tomu musíme uzpůsobit odchylku σ_{mut}, která by měla být menší než v původním návrhu.

Inspirace pro návrh této metody vychází z povahy konkrétní úlohy, není tedy využito žádného obecného přístupu. Účelem této metody generování nových trubek je zejména odolnost vůči omezujícím podmínkám tečné návaznosti oblouků a velikosti středových úhlů. To by mělo umožnit lepší prohledávání prostoru přípustných řešení. Metoda narozdíl od ostatních obecných operátorů pracuje s fenotypem řešení – konkrétní podobou trubek. Obměna pořadí zpracovávání ohybů a jeho uložení v genetické informaci jedince umožňuje identifikovat problematické ohyby.

Při experimentech byly odhadnuty tyto řídící parametry metody:

$$\begin{aligned} \omega &= 2 \\ \sigma_{mut} &= 2mm \\ p_{arcsOrder} &= 0.1 \end{aligned}$$

6.3 Dosažené výsledky

Nyní budeme diskutovat výsledky experimentů s nastavením a vzájemně porovnávat jednotlivé konfigurace NSGA-II z různých hledisek. Podkladem pro hodnocení jsou zejména představené míry kvality aproximace Pareto množiny.

Pro jednotlivé varianty NSGA-II zavedeme následující označení:

Označení	Varianta
REF	Původní návrh algoritmu vycházejícího z NSGA-II (Představen v kapitole 4)
AA	Algoritmus je tolerantnější vůči menší velikosti středových úhlů a zahrnuje toto omezení do definice dominance. Změna je popsána v 6.1.1.
AB	Algoritmus umožňuje uchovávat řešení s trubkami, jejichž počet ohybů je mimo povolené rozmezí. Nedodržení tohoto rozmezí je vyhodnocováno jako porušení omezujících podmínek a zahrnuto do definice dominance. Změna je popsána v 6.1.2.
IP	Rodičovská populace NSGA-II je navýšena oproti interní populaci, takže je vygenerováno více potomků. Změna je popsána v 6.2.1.
AV	Odchylky normálního rozložení pro mutaci genu jsou adaptované v průběhu evoluce. Změna je popsána v 6.2.2.
OVS	Body lomené čáry, jež reprezentuje trubku, jsou podrobeny mutaci v určitém pořadí. Přitom po každé mutaci dochází ke kontrole omezujících podmínek konstrukce trubek. Změna je popsána v 6.2.3.

V příloze B jsou k dispozici výsledky měření pro jednotlivé varianty NSGA-II ve formě, kterou jsme již použili dříve v 5.5. Pro měření statistického rozložení výsledků v deseti simulačních bězích je opět použita střední hodnota a směrodatná odchylka.

Výsledky těchto měření porovnáváme s původním návrhem, tedy variantou REF. Byly pozorovány následující jevy:

- **Diverzita.** Jediná varianta AV dosahuje vyšší diverzity aproximace Pareto fronty u všech testovacích úloh s výjimkou P3B. U variant AB a OVS se diverzita příliš neliší od varianty REF. Varianty AA a IP vykazují horší diverzitu aproximace Pareto fronty zejména na úlohách P3B a P4A.

- **Nalezení přípustného řešení** bylo nejvíce problematické pro úlohu P4A. Varianty AA, AB a IP potřebovaly mnohem více iterací k dosažení jedince vyhovujícího všem omezujícím podmínkám úlohy P4A a také vykazovaly vyšší výkyvy mezi simulačními běhy. Zatímco varianta AV potřebovala stejně někdy i méně iterací a u testovací úlohy P4A vykazovala menší výkyvy. Varianta OVS potřebuje u všech testovacích úloh mnohem více iterací v závislosti na obtížnosti omezujících podmínek.

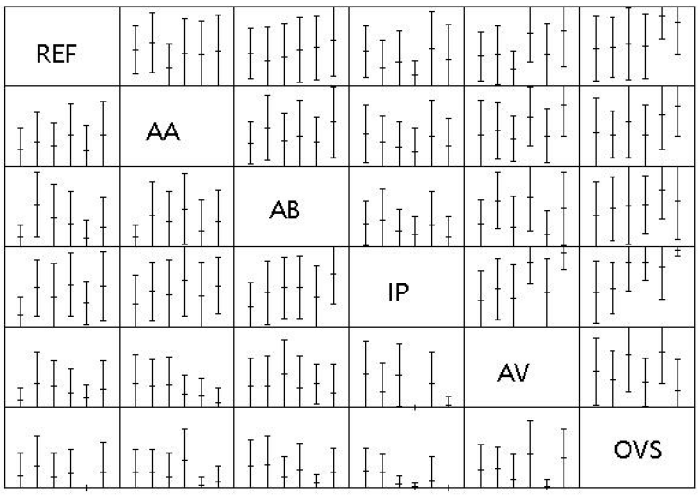
- **Časová náročnost** téměř všech variant je vyšší oproti původnímu návrhu. Pouze v případě varianty IP je časová náročnost srovnatelná, což je dáno povahou experimentu.

V tabulce 6.1 jsou statistické hodnoty míry S jako jediné míry, která sdružuje konvergenci a diverzitu do jedné skalární hodnoty. Opět používáme střední hodnotu a směrodatnou odchylku. OVS dává pro všechny testovací úlohy horší výsledky, zatímco varianta IP pro pět testovacích úloh dává vyšší hodnoty míry S. Ostatní varianty dávají podobné výsledky oproti variantě REF, u některých testovacích úloh lepší, u některých horší. Jednotlivé varianty algoritmu můžeme porovnávat pomocí míry C, jež porovnává dosažené aproximace Pareto množiny z hlediska dominance. Obrázek 6.1 obsahuje grafy statistického rozložení

míry C pro jednotlivé kombinace variant a testovací úlohy. Výsledky jsou zpracovány pomocí střední hodnoty a směrodatné odchylky. Testovací úlohy jsou na ose x uspořádané v obvyklém pořadí: P3A, P3B, P4A, P4B, P6A, P6B. Z grafů lze pozorovat, že právě úlohy s vyšším počtem trubek často rozlišují vlastnosti aproximace Pareto množiny mezi jednotlivými variantami. Nejhorší výsledky z hlediska dominance zřejmě podává varianta OVS, poté varianta AV. Varianta IP poskytuje lepší výsledky na úlohách s vyšším počtem trubek než předchozí varianty. Ostatní varianty nelze z hlediska dominance příliš srovnávat, neboť střední hodnota se často pohybuje kolem 0,5 s vysokou směrodatnou odchylkou.

$S[m^2 \cdot rad]$	P3A	P3B	P4A	P4B	P6A	P6B
REF	2,6469	3,1611	2,2386	2,3218	0,1719	1,8434
	0,0346	0,1986	0,1615	0,0125	0,0075	0,1242
AA	2,6435	3,1909	2,2507	2,321	0,1571	1,8927
	0,0312	0,1472	0,2511	0,0171	0,0106	0,039
AB	2,6259	3,2283	2,1970	2,3260	0,1655	1,8134
	0,0356	0,0923	0,1233	0,0191	0,0144	0,0984
IP	2,6587	3,2563	2,3207	2,3376	0,1582	1,8997
	0,0278	0,0081	0,2166	0,0072	0,0231	0,0234
AV	2,6348	3,128	2,2009	2,2812	0,1612	1,6549
	0,0546	0,2247	0,1569	0,0421	0,0133	0,1434
OVS	2,5890	3,0951	1,9650	2,3086	0,1355	1,6844
	0,0524	0,3207	0,1569	0,0243	0,0188	0,1870

Tabulka 6.1: Míra S pro jednotlivé varianty algoritmu NSGA-II a testovací úlohy.



Obrázek 6.1: Porovnání variant algoritmu pomocí míry C na testovacích úlohách. Řádky a sloupce jsou jednotlivé varianty algoritmu NSGA-II. Pozice (i, j) ukazuje statistické rozložení míry $C(i, j)$ na skupině simulačních běhů pro jednotlivé testovací úlohy.

Na základě získaných výsledků lze usoudit, že zatímco některé varianty (zejména expe-

rimenty s omezujícími podmínkami) vykazují téměř ve všech ohledech horší výsledky, jiné varianty zvyšují úspěšnost algoritmu pouze z jednoho pohledu. Předmětem dalších experimentů by tedy mohly být kombinace jednotlivých variant NSGA-II, například varianta AV s variantou IP, kdy varianta AV dle experimentů zvyšuje diverzitu množiny řešení zatímco varianta IP lépe konverguje k Pareto množině.

Úspěšnost těchto variant by mohl také zvýšit lepší odhad řídících parametrů na základě podrobnější analýzy jejich chování.

Poznamenejme, že interpretace úspěšnosti navržené optimalizační metody závisí na použité sadě testovacích úloh. V testovací sadě se nenachází úlohy s konstantním počtem ohybů. Některé varianty (zejména AB, AV a OVS) by mohly vykazovat lepší výsledky oproti původnímu návrhu právě pro tento typ úloh.

Kapitola 7

Kritické části optimalizace

Nyní analyzujeme kritické části navržené optimalizační metody a nastíníme případná řešení.

Omezený středový úhel ohybu. Co se týče schopnosti metody generovat různá řešení, vykazuje metoda problémy při generování ohybů se středovým úhlem blízkým 180°. Je to zapříčiněno způsobem zakódování trubky pomocí lomené čáry. Tento problém byl analyzován již v návrhu řešení v 4.1. Prostor prohledávaných řešení je tedy omezen. To může být problémem u některých úloh s nedostatečně navrženým rozmezím počtu ohybů na trubku. Tento problém je možné odstranit změnou zakódování řešení. Návrh takového zakódování zřejmě nebude triviální, pokud chceme zachovat výpočty omezujících podmínek a použit stávající variační operátory.

Ohodnocení porušení prostorových omezení. V oblasti ohodnocení porušení prostorových omezení jsme zvolili jednoduchý způsob detekce a ohodnocení kolize mezi trubkami. Opírá se o minimální vzdálenost os trubek. Takovéto ohodnocení vzájemné polohy dvou trubek minimální vzdáleností os trubek ale nebere v úvahu další aspekty (např. objem průniku), které by lépe rozlišovali vzájemnou polohu trubek a umožnili tak lépe mapovat prostor přípustných a nepřípustných řešení optimalizační metodou. Návrh případné modifikace by měl být proveden s ohledem na náročnost řešených úloh a časovou náročnost případných vylepšení. Pro sadu testovacích úloh představených v této práci se vždy podařilo nalézt řešení bez kolizí mezi trubkami, avšak nemusí tomu tak být u složitějších úloh. Podobný přístup je potřeba uplatnit v detekci a ohodnocení kolizí vůči okolním dílům. Jejich přítomnost byla v testovací sadě simulována jednoduchými objekty a rovinou. Přitom jsme opět využívali ve výpočtech vzdálenost k ose trubky a neuvažovali prostorová omezení u konců trubek, kde je potřeba výpočtu vzdálenosti ke kružnici [19].

Nastavení řídicích parametrů. Experimentálně bylo potvrzeno, že úspěšnost evolučního algoritmu závisí na vhodném nastavení řídících parametrů. Třída úloh návrhu výfukových svodů je velice rozmanitá. Odhad řídících parametrů v této práci se opírá zejména o představenou sadu testovacích úloh a některé z řídících parametrů jsou deterministicky odvozeny na základě vstupních parametrů. Návrh takového odhadu pro opravdu rozmanitou sadu úloh může být dost náročný. Adaptace některých řídících parametrů v průběhu evoluce na základě aktuálních měření by mohla tento problém alespoň částečně vyřešit. Při návrhu takového vylepšení je přitom možné využít jiné metody z oblasti softcomputingu (např. *fuzzy regulace*). Další možností je spustit několik simulačních běhů s různým nastavením řídících parametrů či volbou určité varianty. Zde ale může dojít k tomu, že stochastické

efekty zastíní vliv nastavení řídících parametrů, proto je potřeba pro jedno nastavení provést několik restartů algoritmu.

Vliv vstupních parametrů. Použití optimalizační metody v praxi a dosažení pro praktické využití zajímavých výsledků předpokládá vhodné zadání vstupních parametrů – tedy definici pozic trubek na motoru a kolektoru výfuku, poloměr ohybu atd. Pro efektivní práci bude zřejmě potřeba interakce s uživatelem, který bude podle možností upravovat vstupní parametry tak, aby dosáhl lepšího výsledku. Vylepšením aplikace by z tohoto hlediska mohl být automatický návrh změny vstupních parametrů.

Testovací sada úloh. Hodnocení optimalizační metody je závislé na výběru testovacích úloh. Pro kvalitnější zhodnocení metody a jejich kritických míst je vhodné rozšířit testovací sadu zejména o úlohy s obtížnějšími prostorovými omezeními, konkrétní úlohy z praxe, u nichž máme k dispozici nějaké existující řešení získané jinou metodou. To nám pomůže lépe analyzovat použití navrženého systému v praxi. U tohoto typu úloh téměř nikdy neznáme Pareto optimální množinu řešení, takže je problém posoudit kvalitu metody z hlediska globálního prohledávání.

Časová náročnost metody. Jeden simulační běh navrženého evolučního algoritmu trvá obvykle několik minut v závislosti na počtu generací, počtu trubek či náročnosti omezujících podmínek. Pro dosažení kvalitního výsledku je vhodné několikrát algoritmus restartovat. Připomeňme, že detekce a ohodnocení kolizí k okolním dílům bude v praxi zřejmě časově náročnější než dosud simulované prostředí pomocí jednoduchých objektů. Pokud chceme umožnit interakci uživatele s aplikaci tak, aby měl v rozumném čase k dispozici výsledky optimalizace na zadané vstupní parametry, budeme muset algoritmus upravit. Pro takovou optimalizaci bude potřeba analýza časových složitostí pro jednotlivé části algoritmu, na základě nichž určíme kritická místa metody. Předpokládané kritické místo z tohoto pohledu je výpočet omezujících podmínek.

Implementace systému v rámci této práce poskytuje velký prostor pro optimalizaci dílčích algoritmů. Důležitějším požadavkem implementace byla možnost experimentování s metodou. Výpočet je také možné paralelizovat, přičemž evoluční algoritmy nabízí možnost výběru granularity paralelismu v algoritmu. Příkladem je souběžnost více simulačních běhů algoritmu, souběžné ohodnocení jedinců v populaci nebo aplikace paralelních technik na úrovni řadících algoritmů či výpočtu omezujících podmínek.

Kapitola 8

Závěr

Automatický návrh a optimalizace výfukových svodů se v tomto projektu opírá o multikriteriální evoluční algoritmus NSGA-II. Podařilo se navrhnout takový algoritmus, který využívá meta-heuristické techniky evolučních algoritmů a zároveň techniky vycházející z povahy řešené úlohy. Metoda dokáže nalézt řešení vyhovující omezujícím podmínkám pro každou z testovacích úloh. Zároveň ale nemáme prostředky na to, abychom zjistili, zda výsledná nedominovaná množina řešení je opravdu hledanou Pareto optimální množinou.

Pro měření výkonnosti různých MOEA jsou použity míry, které vyhodnocují kvalitu získané nedominované množiny řešení a jsou schopny porovnat dvě získané aproximace Pareto množiny. Na základě těchto měření byly dále navrženy a vyhodnoceny experimenty s nastavením NSGA-II. Varianta se zvětšenou rodičovskou populací má příznivý vliv na konvergenci k Pareto množině, zatímco adaptace odchylek normálního rozložení zvyšuje diverzitu aproximace Pareto množiny. Vzhledem ke stavbě NSGA-II je dále možné experimentovat s metodou a navrhnout tak lepší variantu.

V předchozí kapitole byly identifikovány kritické části optimalizační metody. Nastavení řídících parametrů, ohodnocení kolizí a výběr testovací sady úloh považujeme za nejvíce problematické. Řešení těchto částí mohou být předmětem navazujících projektů.

Pro použití aplikace při návrhu výfukových svodů je nutné zahrnout do výpočtu omezujících podmínek operace s reálnými prostorovými omezeními. Aplikaci je vhodné optimalizovat a integrovat do CAD nástroje, aby mohl uživatel efektivně měnit vstupní parametry a vyhodnocovat výsledky optimalizace. Zároveň by mohlo měření výkonnosti přímo u uživatele poskytnout podklad pro lepší analýzu a odhad řídicích parametrů či návrh vylepšení metody.

Navazující projekty se mohou zaměřit optimalizaci některých vstupních parametrů jako je poloměr ohybu či vhodné pozice jednotlivých trubek na kolektoru výfuku. Užitečným vylepšením systému je také možnost spojení některých trubek před vstupem do výfuku bez použití kolektoru.

Literatura

[1] Bekele, E. G.; Nicklow, J. W.: Multi-objective automatic calibration of SWAT using NSGA-II. *Journal of Hydrology*, ročník 341, č. 3-4, 2007: s. 165 – 176, ISSN 0022-1694, doi:DOI:10.1016/j.jhydrol.2007.05.014.
URL `http://www.sciencedirect.com/science/article/B6V6C-4NTRT8T-1/2/e1ea35c31c0a55a2378bfd3814696721`

[2] Beume, N.: S-metric calculation by considering dominated hypervolume as klee's measure problem. *Evol. Comput.*, ročník 17, č. 4, 2009: s. 477–492, ISSN 1063-6560, doi:http://dx.doi.org/10.1162/evco.2009.17.4.17402.

[3] Bäck, T.; Hammel, U.; Schwefel, H.-P.: Evolutionary Computation: Comments on the History and Current State. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, ročník 1, 1997: s. 3–17.

[4] Chafekar, D.; Xuan, J.; Rasheed, K.: Constrained Multi-Objective Optimization Using Steady State Genetic Algorithms. In *In Proceedings of Genetic and Evolutionary Computation Conference*, Springer-Verlag, 2003, s. 813–824.

[5] Deb, K.; Pratap, A.; Agarwal, S.; aj.: A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, ročník 6, 2000: s. 182–197.

[6] Eiben, A. E.; Smith, J.: *Introduction to Evolutionary Computing*. Hardcover, 2003, iSBN: 978-3-540-40184-1.

[7] Fleischer, M.; Fleischer, M.: The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, Springer, 2003, s. 519–533.

[8] Ho, P. Y.; Shimizu, K.: Simple addition of ranking method for constrained optimization in evolutionary algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, New York, NY, USA: ACM, 2005, ISBN 1-59593-010-8, s. 889–896, doi:http://doi.acm.org/10.1145/1068009.1068158.

[9] Iglesia, B. D. L.; Philpott, M. S.; Bagnall, A. J.; aj.: Data mining rules using multi-objective evolutionary algorithms. In *In Proceedings of 2003 IEEE Congress on Evolutionary Computation*, 2003, s. 1552–1559.

[10] Jadaan, O. A.; Rao, C.; Rajamani, L.: Solving Constrained Multi-objective Optimization Problems Using Non-dominated Ranked Genetic Algorithm. *Asia*

International Conference on Modelling & Simulation, ročník 0, 2009: s. 113–118, doi:http://doi.ieeecomputersociety.org/10.1109/AMS.2009.38.

[11] Jiménez, F.; Verdegay, J. L.: Constrained Multiobjective Optimization by Evolutionary Algorithms. In *University of La*, 1998, s. 266–271.

[12] Kanazaki, M.; Morikawa, M.; Obayashi, S.: Nakahashi: "Multiobjective Design Optimization of Merging Configuration for an Exhaust Manifold of a Car Engine. In *Proc. PPSN, the 7th international*, 2002, s. 281–287.

[13] Laumanns, M.; Thiele, L.; Zitzler, E.: Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers, 2002, s. 439–447.

[14] Mezura-Montes, E.; Coello, C. C.: A survey of constraint-handling techniques based on evolutionary multiobjective optimization. In *Workshop paper at PPSN 2006*.

[15] Naujoks, B.; Beume, N.: Multi-objective Optimisation Using S-metric Selection: Application to three-dimensional Solution Spaces. In *In CEC'2005*, Press, 2005, s. 1282–1289.

[16] PTC: Pro/ENGINEER Behavioral Modeling Extension. `http://www.ptc.com/products/proengineer/behavioral-modeling`, 2010.

[17] Shen, X.; Zhang, M.; Li, T.: A Multi-objective Optimization Evolutionary Algorithm Addressing Diversity Maintenance. In *CSO '09: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, Washington, DC, USA: IEEE Computer Society, 2009, ISBN 978-0-7695-3605-7, s. 524–527, doi:http://dx.doi.org/10.1109/CSO.2009.28.

[18] Vieira, D. A. G.; Adriano, R. L. S.; Krähenbühl, L.; aj.: Handling Constraints As Objectives In A Multiobjective Genetic Based Algorithm. 2002.

[19] Vranek, D.: Fast and accurate circle-circle and circle-line 3D distance computation. *J. Graph. Tools*, ročník 7, č. 1, 2002: s. 23–32, ISSN 1086-7651.

[20] Weise, T.: Global Optimalization Algoritms - Theory and Application. `http://www.it-weise.de`, 2008.

[21] Zitzler, E.; Deb, K.; Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, ročník 8, 2000: s. 173–195.

[22] Zitzler, E.; Giannakoglou, K.; Tsahalis, D.; aj.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm For Multiobjective Optimization. 2002.

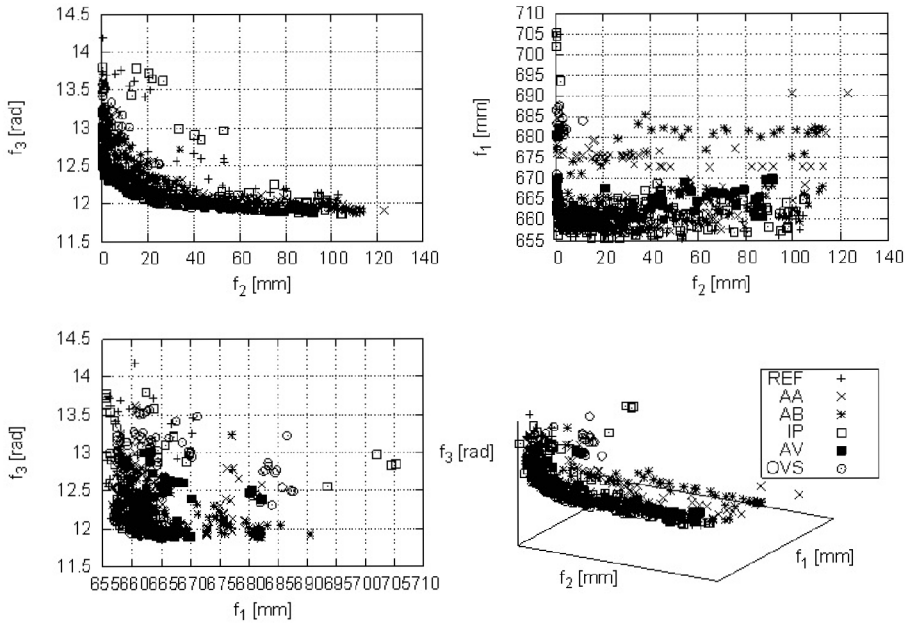
[23] Zitzler, E.; Laumanns, M.; Bleuler, S.: A Tutorial on Evolutionary Multiobjective Optimization. In *In Metaheuristics for Multiobjective Optimisation*, Springer-Verlag, 2003, s. 3–38.

[24] Zitzler, E.; Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. Springer, 1998, s. 292–301.

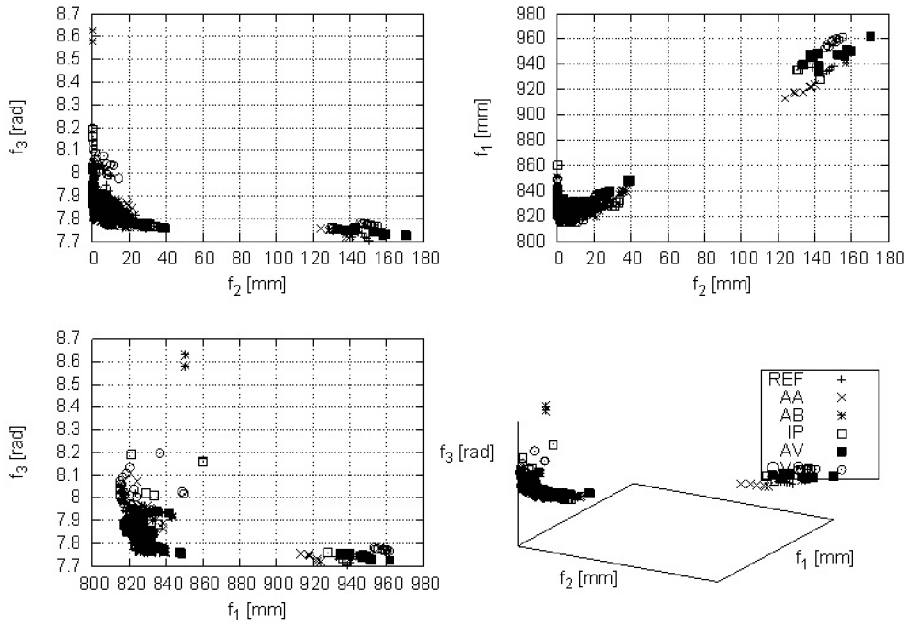
[25] Zitzler, E.; Thiele, L.; Laumanns, M.; aj.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, ročník 7, 2002: s. 117–132.

Dodatek A

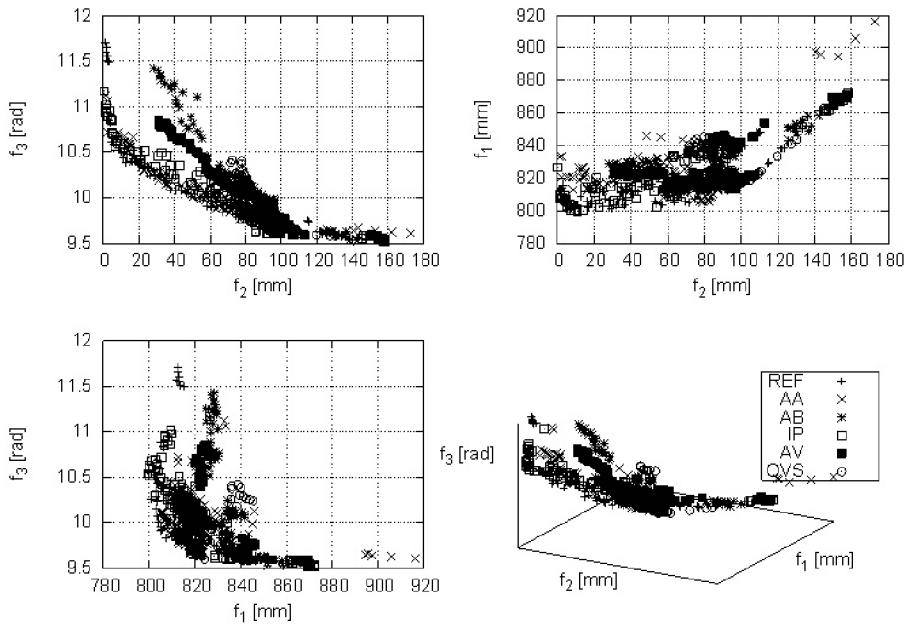
Dosažené aproximace Pareto fronty



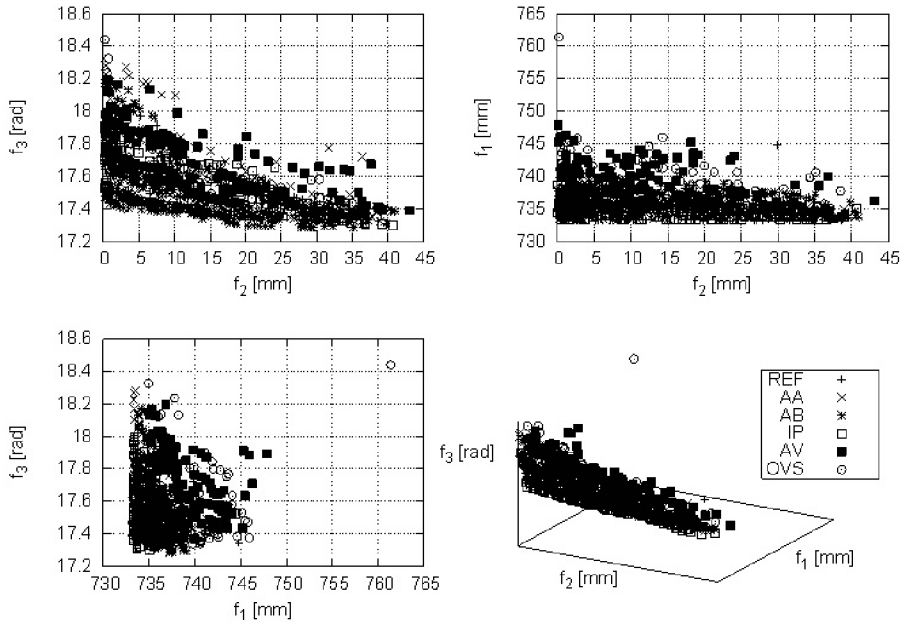
Obrázek A.1: Dosažené aproximace Pareto fronty pro úlohu P3A



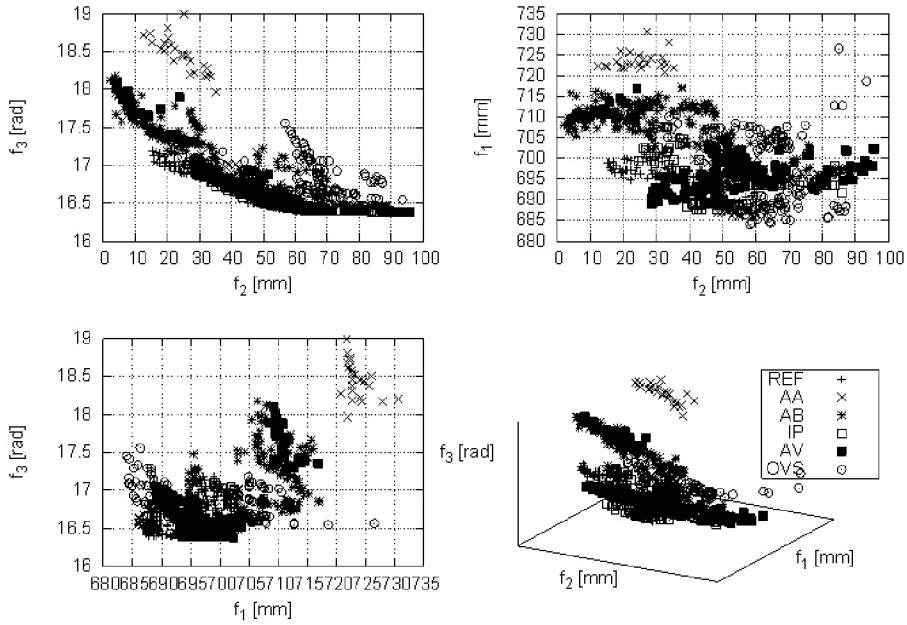
Obrázek A.2: Dosažené aproximace Pareto fronty pro úlohu P3B



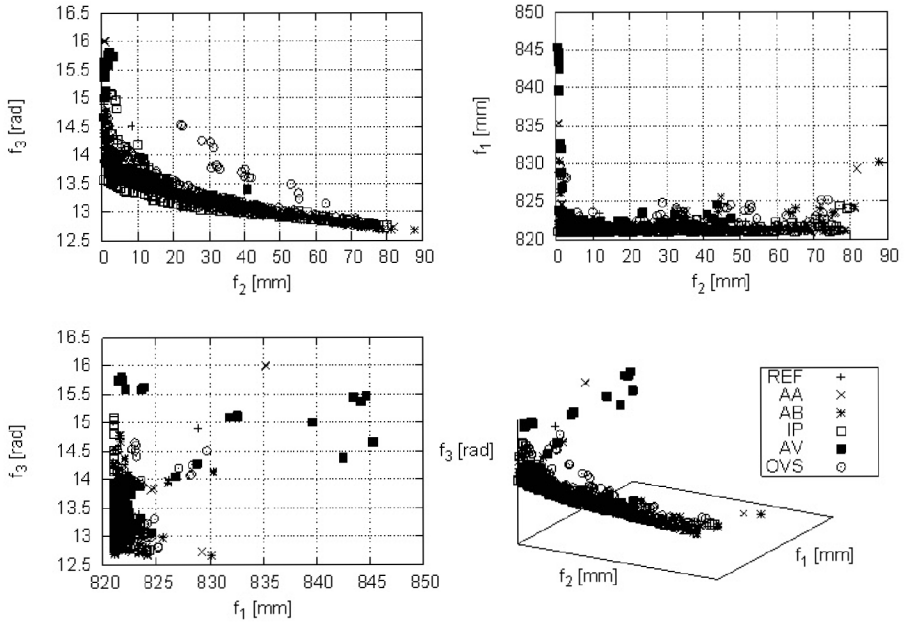
Obrázek A.3: Dosažené aproximace Pareto fronty pro úlohu P4A



Obrázek A.4: Dosažené aproximace Pareto fronty pro úlohu P4B



Obrázek A.5: Dosažené aproximace Pareto fronty pro úlohu P6A



Obrázek A.6: Dosažené aproximace Pareto fronty pro úlohu P6B

Dodatek B

Výsledky měření pro jednotlivé varianty NSGA-II

Míra	**P3A**	**P3B**	**P4A**	**P4B**	**P6A**	**P6B**
$S(P^*)[m^2 \cdot rad]$	2.647	3.161	2.239	2.322	0.1719	1.843
	0.04822	0.2376	0.219	0.02122	0.01003	0.1567
$E(P^*)$	0.9499	0.9277	0.9024	0.9488	0.9158	0.9276
	0.02537	0.02414	0.07987	0.03434	0.0516	0.02582
$\|P^*\|$	53.5	54.5	46.71	60.15	58.58	78.7
	4.188	0.6455	14.39	6.323	13.78	3.132
$\mathcal{I}_{RCGA}$	68.47	38.27	43.34	105	69.22	14.94
	4.397	1.633	2.483	4.022	3.011	0.6443
$F(P_{init})$	0.1401	0.1204	0.1509	0.1467	0.1371	0.1564
	0.005745	0.006449	0.005317	0.003713	0.00304	0.002271
$\mathcal{I}_{NSGA}$	6.643	12.58	33.79	15.77	25.67	22.6
	5.259	2.812	24.54	5.727	7.063	8.499
$F(P_{result})$	0.04963	0.04917	0.02607	0.03073	0.04871	0.04728
	0.01448	0.01074	0.01468	0.01962	0.01469	0.008773
$\mathcal{T}$	50.58	29.66	71.02	105.4	105.5	135.9
	2.514	2.769	7.43	3.129	6.518	6.344

Tabulka B.1: Výsledky měření pro variantu NSGA-II REF

Míra	P3A	P3B	P4A	P4B	P6A	P6B
$S(P^*)[m^2 \cdot rad]$	2.643	3.191	2.251	2.321	0.157	1.893
	0.04748	0.2114	0.2935	0.02128	0.0129	0.04873
$E(P^*)$	0.9465	0.8532	0.8159	0.9421	0.8957	0.9324
	0.02172	0.123	0.1796	0.03452	0.08255	0.04747
$\|P^*\|$	54.1	47.85	45.3	62.64	40.15	77.7
	2.844	8.538	19.78	5.709	13.52	1.735
$\mathcal{I}_{RCGA}$	68.04	37.08	13.93	101.6	67.52	11.21
	2.384	1.297	0.459	2.814	1.556	0.5109
$F(P_{init})$	0.1406	0.1236	0.1599	0.1458	0.1434	0.1605
	0.004292	0.00647	0.008199	0.00395	0.004575	0.002208
$\mathcal{I}_{NSGA}$	6.6	12.08	48.1	16.09	34.85	27.2
	2.458	2.841	36.88	3.397	19.49	8.998
$F(P_{result})$	0.05452	0.03979	0.03752	0.03372	0.04019	0.04796
	0.01615	0.01388	0.02073	0.01404	0.01134	0.0105
$\mathcal{T}$	52.74	29.48	74.43	136.3	112.1	147.4
	2.703	3.892	5.747	4.843	8.008	5.494

Tabulka B.2: Výsledky měření pro variantu NSGA-II AA

Míra	P3A	P3B	P4A	P4B	P6A	P6B
$S(P^*)[m^2 \cdot rad]$	2.626	3.228	2.197	2.326	0.1655	1.813
	0.04677	0.1555	0.1776	0.02464	0.01706	0.142
$E(P^*)$	0.9405	0.9236	0.9271	0.9457	0.9134	0.9334
	0.02973	0.02976	0.05372	0.02639	0.0568	0.04006
$\|P^*\|$	53.8	55.2	45.9	60.8	59.5	80
	2.315	2.638	17.6	3.868	18.83	2.098
$\mathcal{I}_{RCGA}$	72.66	39.87	47.81	124.2	77.05	16.75
	4.288	2.644	3.563	5.115	3.139	0.5377
$F(P_{init})$	0.1147	0.1011	0.1039	0.1241	0.09657	0.09842
	0.008063	0.005054	0.00262	0.005038	0.003429	0.002712
$\mathcal{I}_{NSGA}$	6.9	12.1	50	14.8	32.7	25.9
	2.879	3.859	42.47	3.458	7.239	12.04
$F(P_{result})$	0.0417	0.03896	0.01997	0.01508	0.02875	0.0299
	0.01041	0.01518	0.01303	0.008362	0.009729	0.006494
$\mathcal{T}$	54.6	30.51	78.41	153.8	122.6	144.2
	1.639	2.597	6.42	7.259	12.73	12.78

Tabulka B.3: Výsledky měření pro variantu NSGA-II AB

Míra	**P3A**	**P3B**	**P4A**	**P4B**	**P6A**	**P6B**
$S(P^*)[m^2 \cdot rad]$	2.659	3.256	2.321	2.338	0.1582	1.9
	0.03709	0.1359	0.2732	0.009054	0.03545	0.03182
$E(P^*)$	0.9486	0.8898	0.8657	0.9189	0.9027	0.9209
	0.02427	0.05288	0.09591	0.04623	0.0553	0.02288
$\|P^*\|$	55.6	49.3	42.78	63.9	50.3	78.3
	1.855	7.029	15.69	2.663	17.24	3.689
$\mathcal{I}_{RCGA}$	69.32	38.15	43.25	105.3	68.33	14.68
	1.61	1.535	4.418	2.785	1.347	0.5468
$F(P_{init})$	0.1429	0.1204	0.1553	0.1453	0.1409	0.1557
	0.005367	0.006252	0.003669	0.00333	0.006074	0.006392
$\mathcal{I}_{NSGA}$	5.3	9.9	43.89	12.6	29	19.3
	2.283	2.256	62	3.527	16.58	6.604
$F(P_{result})$	0.05599	0.03978	0.03753	0.03533	0.04644	0.06425
	0.00999	0.009662	0.01977	0.01398	0.00953	0.01355
$\mathcal{T}$	48.24	23.62	71.73	132.4	98.53	121.9
	1.885	2.84	6.162	2.701	3.941	5.382

Tabulka B.4: Výsledky měření pro variantu NSGA-II IP

Míra	**P3A**	**P3B**	**P4A**	**P4B**	**P6A**	**P6B**
$S(P^*)[m^2 \cdot rad]$	2.635	3.128	2.201	2.281	0.1612	1.655
	0.06677	0.306	0.2121	0.06939	0.01553	0.1544
$E(P^*)$	0.9778	0.8788	0.9399	0.9674	0.9762	0.9789
	0.01824	0.1176	0.05213	0.03509	0.02235	0.01198
$\|P^*\|$	53.6	49.1	48.67	62.5	62.8	75.4
	2.973	8.734	18.13	1.118	16.04	10.29
$\mathcal{I}_{RCGA}$	70.18	37.91	44.28	103.9	68.96	14.92
	3.998	2.47	3.576	4.425	1.599	0.8926
$F(P_{init})$	0.1382	0.123	0.1501	0.1451	0.1411	0.1544
	0.007451	0.007137	0.007407	0.00391	0.003956	0.003921
$\mathcal{I}_{NSGA}$	7.9	7.5	36.11	23.1	24.7	20.3
	4.867	1.688	19.5	10.96	9.209	5.728
$F(P_{result})$	0.04116	0.0486	0.02877	0.01529	0.03741	0.05118
	0.02005	0.01756	0.0161	0.01652	0.01809	0.01302
$\mathcal{T}$	58.41	32.68	81.63	156.9	125.8	152
	3.724	3.409	8.563	4.801	15.5	14.12

Tabulka B.5: Výsledky měření pro variantu NSGA-II AV

Míra	**P3A**	**P3B**	**P4A**	**P4B**	**P6A**	**P6B**
$S(P^*)[m^2 \cdot rad]$	2.589	3.095	1.965	2.309	0.1355	1.684
	0.06399	0.4079	0.1962	0.029	0.02363	0.2443
$E(P^*)$	0.9321	0.9322	0.9242	0.9631	0.9203	0.9653
	0.03913	0.04432	0.07995	0.03177	0.07445	0.02139
$\|P^*\|$	51.6	47.5	26.8	61.3	37.8	75.4
	4.716	9.211	15.85	7.267	14.8	5.535
$\mathcal{I}_{RCGA}$	68.35	38.23	44.18	104.8	69.35	15.04
	2.385	1.323	3.309	3.651	2.883	0.4698
$F(P_{init})$	0.1378	0.125	0.1511	0.1454	0.1404	0.1565
	0.004424	0.007562	0.008546	0.004135	0.003572	0.005686
$\mathcal{I}_{NSGA}$	7	27	109.2	24.9	35.3	52
	4.29	7.589	89.87	12.96	12.71	20.14
$F(P_{result})$	0.02297	0.02846	0.0143	0.006936	0.0286	0.03312
	0.007626	0.01918	0.007412	0.003972	0.01281	0.007964
$\mathcal{T}$	51.19	29.63	72.75	138.2	113.7	145.8
	2.648	3.384	2.55	3.786	7.132	8.909

Tabulka B.6: Výsledky měření pro variantu NSGA-II OVS