VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

Evoluční výpočetní techniky

Evolutionary computing techniques

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Matěj Goněc
AUTHOR

VEDOUCÍ PRÁCE — RNDr. Jiří Dvořák, CSc.
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Matěj Goněc**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**RNDr. Jiří Dvořák, CSc.**
Akademický rok:	2023/24

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

Evoluční výpočetní techniky

Stručná charakteristika problematiky úkolu:

Evoluční výpočetní techniky se používají k řešení složitých optimalizačních problémů. Zajímavé jsou např. evoluční metody inspirované přírodními procesy (genetické algoritmy, mravenčí algoritmy, hejnové algoritmy) nebo technickými procesy (simulované žíhání).

Cíle bakalářské práce:

1. Charakterizovat problematiku evolučních optimalizačních výpočtů.
2. Popsat vybrané metody s uvedením příkladů jejich aplikace a provést jejich srovnání.

Seznam doporučené literatury:

ZELINKA, I. a kol. Evoluční výpočetní techniky - principy a aplikace. BEN, Praha, 2009.

SÖRENSEN, K. Metaheuristics – the metaphor exposed. International Transactions in Operational Research, vol. 22, issue 1, 2015, pp. 3-18.

RAJPUROHIT, J. et al. Glossary of metaheuristic algorithms. International Journal of Computer Information Systems and Industrial Management Applications, vol. 9, 2017, pp. 181-205.

SLOWIK, A., KWASNICKA, H. Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, vol. 32, 2020, pp. 12363-12379.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

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

ABSTRAKT

Tato práce se zabývá evolučními výpočetními technikami. Konkrétně popisuje použití metod genetických algoritmů a diferenciální evoluce na problém hledání cesty mobilního robota.

ABSTRACT

This thesis deals with evolutionary computing techniques. Specifically, it describes the application of genetic algorithm and differential evolution to path planning of mobile robot.

KLÍČOVÁ SLOVA

Optimalizace, evoluční výpočetní techniky, genetický algoritmus, diferenciální evoluce.

KEYWORDS

Optimization, evolutionary computing techniques, genetic algorithm, differential evolution.

ÚSTAV AUTOMATIZACE A INFORMATIKY

2024

BIBLIOGRAFICKÁ CITACE

GONĚC, Matěj. *Evoluční výpočetní techniky*. Brno, 2024. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/154095. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce: RNDr. Jiří Dvořák, CSc.

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu práce RNDr. Jiřímu Dvořákovi, CSc. za jeho cenné rady k této práci.

ČESTNÉ PROHLÁŠENÍ

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

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

V Brně dne 20. 5. 2024 .....

Matěj Goněc

OBSAH

1 ÚVOD

Tato práce se zabývá evolučními výpočetními technikami (EVT), což jsou moderní algoritmy, které se s oblibou používají pro řešení náročných inženýrských problémů současné doby. Tyto algoritmy nacházejí inspiraci v základních přírodních principech a z velké částí jsou na nich také postaveny. Za ty nejzásadnější přírodní principy pro EVT se považují Darwinova a Mendelova teorie evoluce, ze kterých přebírají myšlenku přenosu genetické informace z rodičů na potomky. Stěžejní částí EVT jsou tzv. evoluční algoritmy, ale jsou v nich zahrnuty například i algoritmy inspirované inteligencí hejna, či další techniky jako genetické programování nebo evoluční hardware. Tato práce se soustředí na evoluční algoritmy, konkrétně to bude genetický algoritmus a diferenciální evoluce.

V úvodu práce bude nastíněna problematika optimalizace, pod kterou spadají EVT. Dále budou uvedeny hlavní přednosti EVT, díky kterým jsou s úspěchem aplikovány na problémy, které nejsme se současným stavem technologií řešit deterministicky. To znamená, že nejsme schopni kvůli výpočetní náročnosti najít nejlepší řešení. EVT nám v takovýchto situacích dodají sice ne právě nejlepší čili optimální řešení, ale řešení jemu velmi blízké. Úvodní kapitola bude také obsahovat různé druhy dělení EVT.

Následně pak budou popsány teoretické základy dvou konkrétních algoritmů, které jsou předmětem této práce. Těmito algoritmy jsou: genetický algoritmus a diferenciální evoluce. Tyto algoritmy se používají pro řešení velkého rozsahu soudobých problémů, zde budou aplikovány na problém hledání cesty mobilního robota. Popisu tohoto problému bude věnována samostatná kapitola práce.

V praktické části bude popsáno vlastní řešení implementace výše uvedených algoritmů na daný problém. V závěrečné části práce budou porovnány výsledky dosažené pomocí jednotlivých algoritmů.

2 EVOLUČNÍ VÝPOČETNÍ TECHNIKY

2.1 Optimalizace

Než se dostaneme k evolučním výpočetním technikám (dále jen EVT), je potřeba začít u pojmu optimalizace, který je nadřazený EVT. Optimalizace je proces, při němž se snažíme nalézt výhodné a efektivní řešení problému. Bavíme se tedy o tzv. optimalizačních úlohách, které popisují problémy vznikající při každodenní inženýrské praxi. Nejstarší úlohy se datují až do období antiky, kde byla snaha o nalezení extrémů různých veličin. Z historie se dochovala spousta dnes již známých optimalizačních úloh, například Didonina úloha, ve které fénická královna dostala darem kus země, kterou by dokázala vymezit pomocí dobytčí kůže. V legendě je úloha vyřešena pomocí tenkých proužků kůže. K rozvoji optimalizace také přispěly osobnosti jako Isaac Newton nebo Daniel Bernoulli pomocí zmínek ve svých knihách. Obrovský rozmach nastal po druhé světové válce v odvětví ekonomie. S úspěchem se optimalizovaly např problémy rozvrhování výroby. V dnešní době nalézá optimalizace uplatnění ve velkém množství odvětví, od chemie přes kybernetiku až po technické konstruování ve stavebnictví, loďařství či raketovém inženýrství.

Hledání řešení praktických problémů vede k výpočtu extrémů funkcí. Taková funkce, u které hledáme její maximální nebo minimální hodnotu, se nazývá účelová funkce. Historicky je také nazývána jako „cenová funkce“, protože pomocí této funkce dokážeme ocenit, jinak řečeno ohodnotit, nalezené řešení problému dosazením do argumentů funkce. Nejčastěji se tato funkce označuje $f_{cost}(x)$, anglické slovo *cost* se zde překládá jako cena.

Tvorba účelové funkce je kritický krok při definování optimalizačního modelu, což je označení pro přepis praktického problému do matematické podoby, kterou pak můžeme optimalizovat. Navzdory důležitosti tohoto kroku a také vzhledem k rozmanitosti řešených problémů neexistuje univerzální návod nebo příručka, jak tento krok zvládnout. Samotná tvorba je totiž velmi úzce spjatá s odbornými znalostmi z oblasti řešeného problému, a tak je ve většině případů potřeba spolupráce více lidí [1].

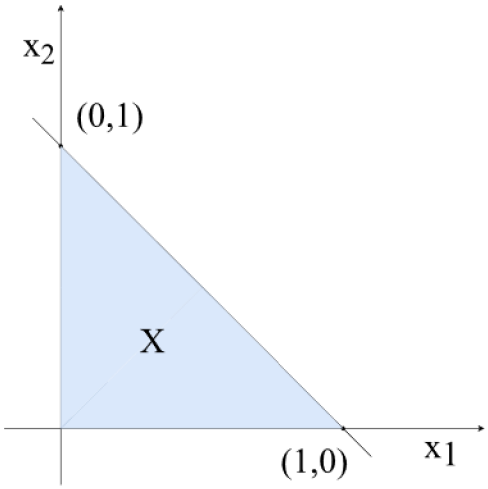
Dále potřebujeme pro optimalizační model znát omezení, která nám, jak název naznačuje, omezují prohledávaný prostor, a to buď z hlediska smysluplnosti nebo různých požadavků. Souhrn všech takovýchto omezení nám vydefinuje množinu přípustných řešení, nechť je tato množina označena jako X. V rovnici (1) pak můžeme vidět, jak je definován základní optimalizační problém pomocí obecné účelové funkce a omezení neboli podmínek.

$$Minimalizuj\ f_{cost}\ (\vec{\mathbf{x}})\ \ za\ podmínek\ \vec{\mathbf{x}}\ \in\ X \tag{1}$$

Pro lepší pochopení problematiky je uveden konkrétní optimalizační problém převzatý z [2]. V rámci tohoto problému bude graficky objasněn pojem množiny přípustných řešení pomocí obrázku 1. Nejdříve je však uveden matematický zápis problému (2).

$$\begin{aligned} &Minimalizuj\ f(x_1, x_2) \\ &za\ podmínek \quad x_1 \geq 0 \\ &\qquad\qquad\quad x_2 \geq 0 \\ &\qquad\qquad\quad x_1 + x_2 \leq 1 \end{aligned} \tag{2}$$

Omezení jsou typicky vyjádřena pomocí rovnic a nerovnic, to poznáme podle použití matematických znaků jako jsou ≤, ≥ a =. Při výskytu < nebo >, kterým říkáme ostré nerovnosti, hranice omezení nespadá do přípustné množiny.



Obr. 1: Množina přípustných řešení uvedeného problému [2]

Prohledávaný prostor na obrázku 1 je poměrně jednoduchý. Člověk, který se už setkal s reálnými inženýrskými problémy, však ví, že zdaleka ne všechny problémy mají takto jednoduchý prohledávaný prostor. Některé problémy se vyznačují velkou složitostí prohledávaného prostoru, která je dána především velkým počtem lokálních extrémů a komplikovanými omezujícími podmínkami. U takto složitých prohledávaných prostorů nám výpočetní nároky neumožňují najít nejlepší řešení. Právě zde přicházejí vhod EVT, které si jsou schopné s takto složitými problémy poradit.

2.2 Evoluční výpočetní techniky

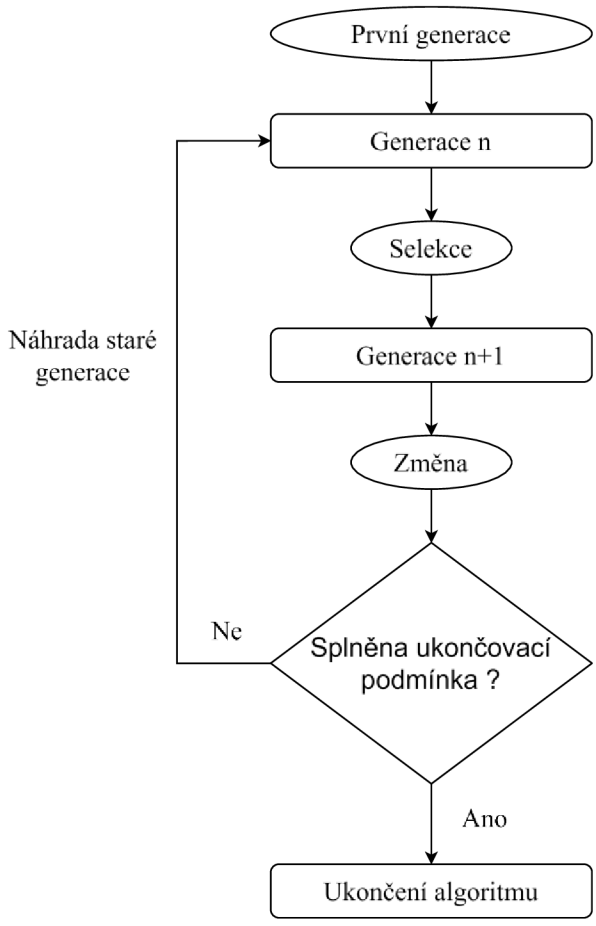
V dnešní prudce se rozvíjející době je potřeba řešit stále větší a náročnější optimalizační problémy, na které přestávají stačit základní optimalizační metody. Je opravdu snadné se v rámci dnešních inženýrských problémů dostat do situace, kdy je časová složitost optimalizačního algoritmu taková, že doba pro nalezení nejlepšího řešení by byla delší než známá doba existence vesmíru. Z tohoto důvodu se tedy v oblasti optimalizace hledají možnosti, jak řešit takovéto druhy problémů.

Zásadní inspirace byla nalezena v přírodě. Zmínky o inspiraci přírodou, byť ne úmyslné, se objevují už v první polovině 60. let, kdy skupinu berlínských studentů

napadlo při řešení konstrukce převodovky náhodně kombinovat již existující konstrukční řešení v úsilí zlepšit jednotlivé parametry převodovky [3].

Samotný název evoluční výpočetní techniky (dále jen EVT) je tedy pojem zahrnující oblast speciálních technických výpočtů inspirovaných přírodními vývojovými procesy a jevy. Tak jako tomu je v přírodě, tak i u EVT hraje svou podstatnou roli i náhoda, která má zásadní vliv na vlastnosti těchto technik a také na výsledky, které z nich dostáváme. [3]

První zásadní charakteristika EVT je skutečnost práce s populací, což je množina řešení (jedinců), se kterou se pracuje a která je postupně zlepšována. Každé jednotlivé řešení z této populace má své ohodnocení podle předem určené cenové (účelové) funkce. Dále je simulován přírodní proces výběru nejlepších jedinců (selekce) z dané populace. Poté přichází na řadu klíčový prvek EVT a tou je změna. Existují dva základní druhy takovéto změny a to jsou: Mutace, ta spočívá ve vytvoření nového jedince malou mutací jedince ze stávající populace. Křížení, zde nový jedinec vzniká křížením dvou nebo více rodičů. Nový jedinec (potomek) potom obsahuje část z každého svého rodiče. Vzniká nová generace jedinců, která nahrazuje předchozí generaci. První generace je většinou náhodně zvolená množina řešení, avšak nevylučuje se zakomponování dříve zjištěné vlastnosti o řešení, což má pozitivní vliv na řešení. Celý proces zachycuje diagram na obrázku 4 [3].



Obr. 2: Diagram EVT inspirovaný [3]

Oproti klasickým deterministickým algoritmům se EVT řadí mezi tzv. heuristiky, což jsou optimalizační metody dodávající přibližné ale uspokojivé výsledky tam, kde klasické deterministické metody nejsou schopny najít řešení v rozumném čase [4].

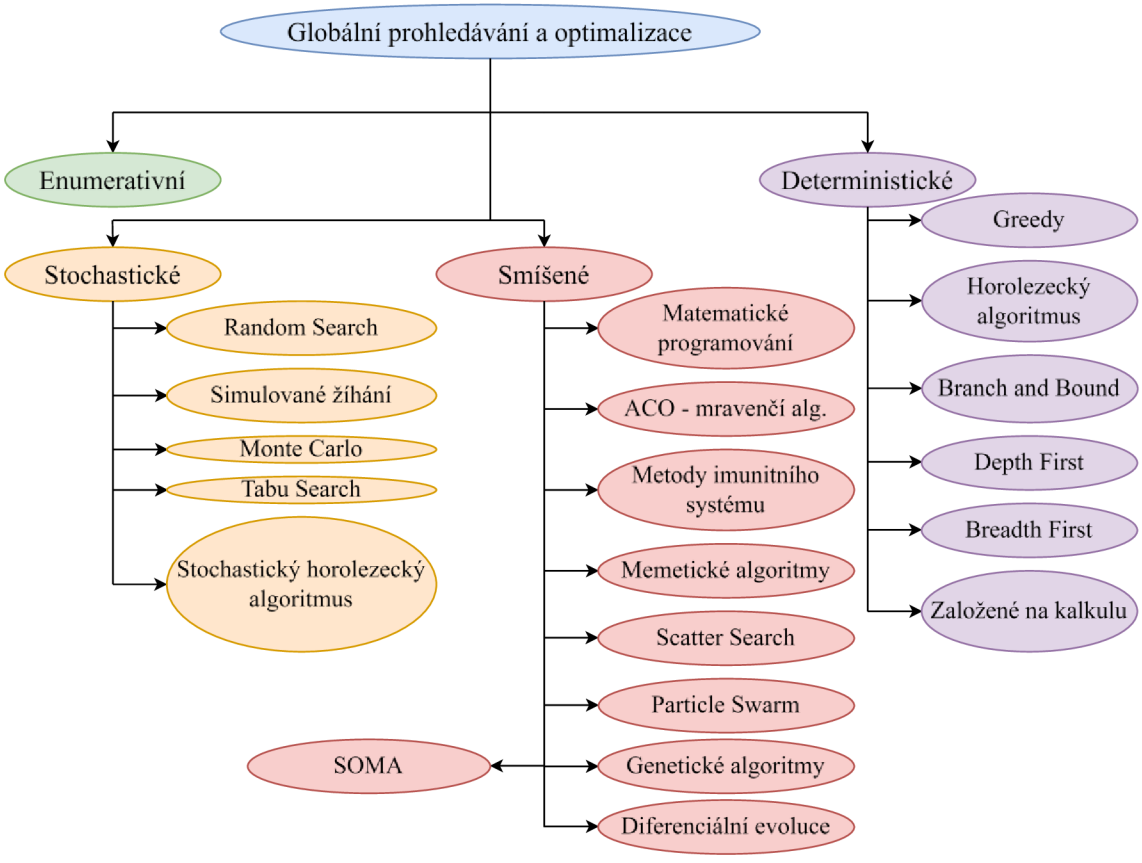
EVT nacházejí uplatnění v mnoha inženýrských odvětvích od elektroniky přes umělou inteligenci až po automatizaci a stávají se velmi populárním nástrojem. Tento fakt je podložen daty z praxe, kde bylo zjištěno, že v minulém desetiletí bylo na vědeckých webových stránkách jako jsou například *IEEE Xplore, Science Direct* nebo *Web of Science* publikováno okolo 1 800 000 článků, které se zmiňovaly o EVT [5]. Toto číslo však může být poněkud zavádějící, vzhledem ke snaze vytvořit algoritmus inspirovaný kdečím. Dají se dohledat algoritmy inspirované od fotbalového turnaje *FIFA World Cup* [6], kde se využívá metoda soutěže, přes *Water Cycle Algorithm* [7], který optimalizuje pomocí myšlenky přirozeného oběhu vody na planetě Zemi, až po *Zombie Survival Optimization* [8], inspirovaný zombie apokalypsou. Navzdory kreativitě nemusí být všechny takovéto algoritmy použitelné.

2.3 Rozdělení optimalizačních a evolučních výpočetních technik

Současná doba představuje obrovskou škálu problémů, které je potřeba řešit a které se dají řešit optimalizačními algoritmy. Tato škála je velmi pestrá a neustále se rozrůstá, naštěstí se stejné rozmanitosti dostává i optimalizačním algoritmům, přičemž jsou stále hledány nové a efektivnější optimalizační metody. Cílem této kapitoly je tedy nejprve představení rozdělení optimalizačních technik podle jejich charakteristických vlastností a zařazení EVT v něm. Dále bude zmíněno něco i o rozdělení samotných EVT. Nejprve nechť je tedy uvedeno rozdělení optimalizačních algoritmů podle jejich vlastností [9]:

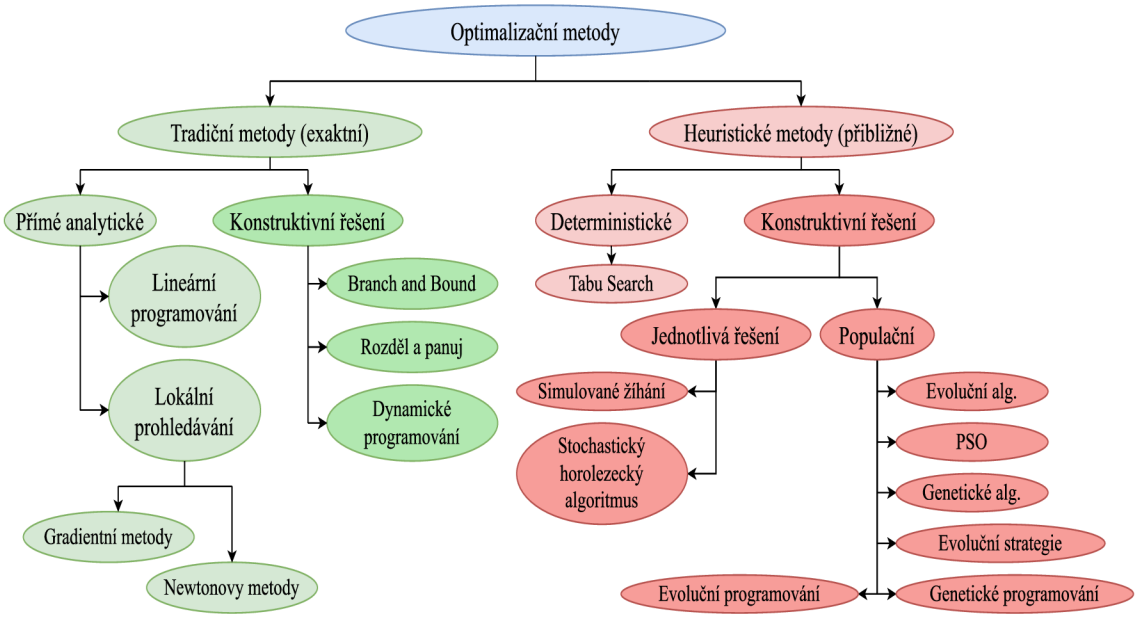
- **Enumerativní.** Je to kategorie takových algoritmů, které projdou celý prostor přípustných řešení a vyberou z něj to nejlepší řešení. Hodí se tedy pro problémy, kde je prohledávaný prostor řešení neobjemný. Pokud by byl použit tam, kde je prohledávaný prostor příliš velký, mohla by jednoduše nastat situace, kdy by algoritmus této kategorie potřeboval na dokončení výpočtu větší čas, než je známá doba existence vesmíru.
- **Deterministické.** Tyto algoritmy pracují na základě jasně definovaných matematických principů, které jsou postavené na důkazech. Vyžadují také jisté matematické předpoklady ohledně cenové funkce a prohledávaného prostoru. Jelikož rozdělení je uvedeno pro vytvoření základního přehledu, nebudou tyto omezující předpoklady dále rozebírány. Algoritmy této skupiny se také vyznačují tím, že při různých běžích dodávají stále stejný výsledek.

- **Stochastické.** Tyto algoritmy se vyznačují využitím náhody. Je to v podstatě náhodné prohledávání a jako výsledek dostaneme nejlepší řešení nalezené při jednom určitém spuštění algoritmu. Kvůli výskytu náhody se ale nejlepší nalezená řešení mohou při různých spuštěních lišit. Je proto vhodné tyto algoritmy pustit vícekrát a vybrat nejlepší řešení ze všech spuštění. Jsou ale většinou pomalé a nezajišťují nalezení obstojného výsledku, proto se dostáváme k poslední skupině.
- **Smíšené.** Poslední skupina algoritmů je, jak její název naznačuje, důmyslnou směsí algoritmů deterministických a stochastických, které při své spolupráci dodávají překvapivě dobré výsledky. Právě do této skupiny spadají evoluční algoritmy, jež jsou předmětem této práce. Ani jedna z předchozích kategorií není příliš vhodná pro optimalizace se složitým prohledávaným prostorem, tato skupina je řešením tohoto problému.



Obr. 3: Rozdělení optimalizačních algoritmů s jednotlivými příklady, překresleno z [9]

Je však nezbytné podotknout, že popsané rozdělení není jediné možné. Univerzální rozdělení, které by bylo uznáno většinou, se stále hledá. Na obrázku 3 je předvedeno toto rozdělení i s jednotlivými zástupci zmíněných skupin optimalizačních algoritmů. Přední vědci v této oblasti se stále dohadují o příslušnosti některých algoritmů do daných skupin. Aby byla lépe pochopitelná rozmanitost těchto rozdělení, bude na obrázku 4 představen jiný způsob rozdělení optimalizačních algoritmů.



Obr. 4: Rozdělení optimalizačních metod podle [10]

Nyní něco k rozdělení samotných EVT. Jak již bylo zmíněno výše, EVT jsou pestrý a objemný pojem. V současných textech se pod EVT řadí 5 základních technik: genetické algoritmy, genetické programování, diferenciální evoluce, evoluční strategie a evoluční programování [5]. Samostatným podoborem jsou potom algoritmy inspirované kolektivním chováním živočichů. Jsou to algoritmy založené na principu inteligence hejna. Odborníci se rozchází v názoru, jestli se hejnové algoritmy řadí pod EVT nebo by měly představovat samostatnou skupinu.

Genetické programování je speciální druh genetického algoritmu, který byl relativně nedávno vynalezen jako pokus o automatické generování programového kódu. Evoluční strategie se vyznačují tím, že místo vytvoření nové generace pomocí rodičů, vzniká dočasná generace. V ní proběhnou mutace a křížení, poté se vybere požadovaný počet nejlepších řešení z této generace do té nové [5]. Evoluční programování spočívá ve dvou zásadních krocích: první je mutace řešení v současné populaci a druhý je následný výběr nejlepších řešení ze současné populace a zmutovaných řešení [11].

Jelikož jsou hlavním předmětem této práce, budou genetickým algoritmům a také diferenciální evoluci věnovány samostatné kapitoly.

2.4 Genetické algoritmy

Původní verze genetických algoritmů jsou nejstarším a také nejznámějším zástupcem EVT. Jejich název vychází z jejich inspirace přírodními procesy evoluce popsanými Charlesem Darwinem a J. G. Mendelem. Mluví se o původní verzi genetických algoritmů, protože jako nejstarší zástupce EVT pochází z 80. let minulého století. V současné době samozřejmě došlo k vývoji a úpravám genetických algoritmů, tak aby byly použitelné a zároveň efektivnější ve větší škále problémů, ale o tom později.

Díky své inspiraci přírodními procesy je i terminologie analogická s tou biologickou. Základním prvkem genetických algoritmů je parametr jedince, kterému se říká gen, ten se typicky pro původní verze genetických algoritmů kódoval binárně. Soubor těchto parametrů (genů) je nazván chromozóm a ten pak reprezentuje každého jedince. Chromozóm má typicky pro původní verze fixní délku a dále se s chromozomy pak nakládá pomocí genetických operací.

2.4.1 Počáteční populace

Na začátku celého procesu stojí počáteční populace. Ta se většinou získá náhodným generováním jednotlivých chromozomů, ale není také vyloučeno použití předchozích znalostí problému, či využití vhodných řešení získaných jinými heuristickými metodami. To pak má pozitivní vliv na celý průběh hledání řešení. Dále je zde nutnost zvolení vhodného počtu jedinců v počáteční populaci. Tady se střetávají dva přístupy. Jeden přístup je zvolení co největšího počtu, což přináší značnou rozmanitost počátečních jedinců a tím zvyšuje šanci nalezení optimálního řešení. Naopak zvolení menšího počtu nabízí rychlejší konvergenci algoritmu, což znamená že dostaneme uspokojivé řešení v kratším čase, avšak je větší pravděpodobnost, že se zastavíme například v lokálním minimu.

2.4.2 Vhodnost jedince

U každého jedince populace je posouzena jeho vhodnost pro řešení problému. Toto se většinou děje pomocí cenové funkce. Hodnota vhodnosti je každému jedinci přidělena buďto přepočtem cenové (účelové) funkce anebo hodnotu vhodnosti představuje přímo hodnota cenové funkce u každého jedince. Takto přidělená hodnota vhodnosti (v anglické literatuře fitness) je rozhodující při selekci vhodných rodičů a následné reprodukci. Úroveň vhodnosti může být také důležitá pro ukončení celého procesu.

2.4.3 Selekce rodičů

Po ohodnocení každého jedince populace proběhne výběr rodičů pro reprodukci, tedy vytvoření potomků pro novou populaci. Tito potomci budou ještě upraveni pomocí genetických operátorů. Základní způsoby výběru rodičů jsou [12]:

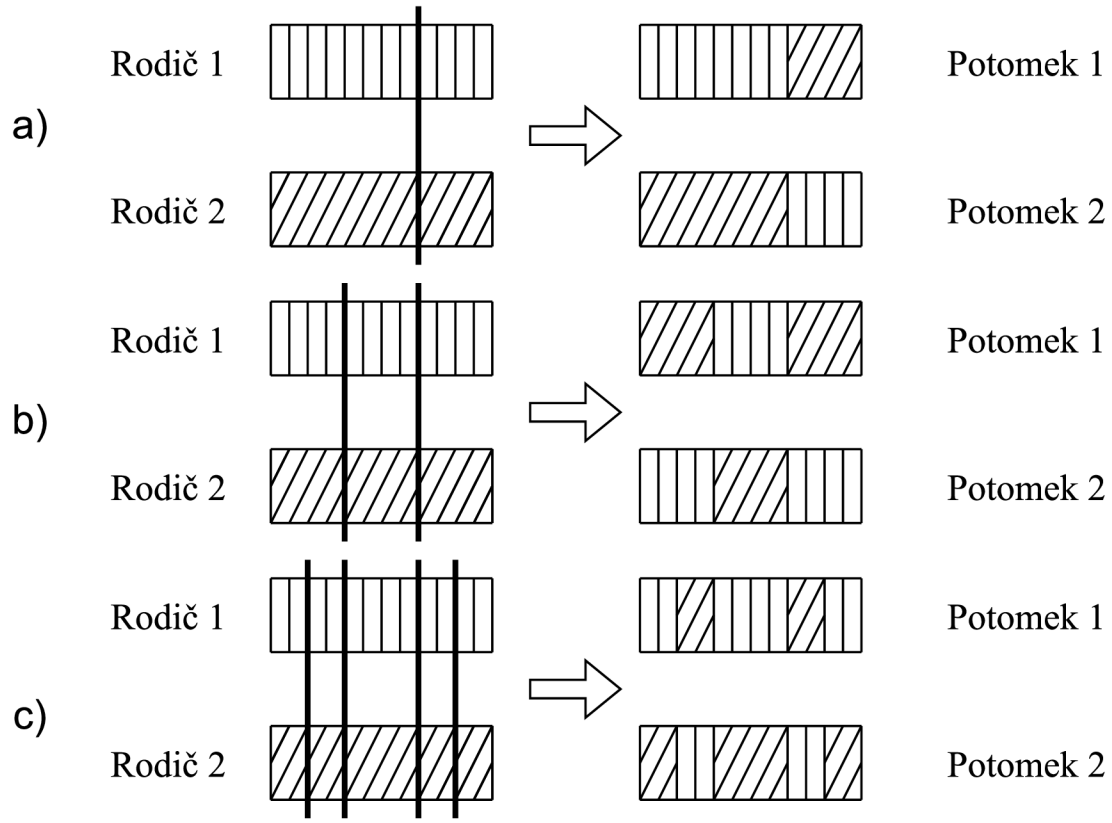
- **Turnajový způsob**. K výběru tímto způsobem dochází pomocí náhodného výběru několika jedinců a následného porovnání jejich vhodností. Jako rodiče jsou vybráni ti s nejlepšími předpoklady.
- **Proporcionální způsob**. Tento způsob je založen na přidělení pravděpodobnosti výběru pro rodičovství každému jedinci z populace. Čím je větší hodnota vhodnosti daného jedince, tím vyšší je pravděpodobnost, že bude vybrán. Samotný výběr se poté provádí pomocí ruletového systému, kdy je jedinci přidělena určitá výseč kola rulety. Dále je náhodně vygenerováno číslo od 0 do 1. Podle výseče, do které číslo spadá, je vybrán jedinec, který náleží této výseči.
- **Uspořádaný výběr**. Uspořádaný výběr je velmi podobný proporcionálnímu způsobu, kdy jsou také přiděleny všem jedincům pravděpodobnosti podle jejich

vhodnosti. Jedinec je pak vybrán náhodně bez ruletového systému. Samozřejmě s přihlédnutím k obdrženým pravděpodobnostem.

Po výběru rodičů probíhá reprodukce pomocí genetických operátorů.

2.4.4 Genetické operátory

Základní genetické algoritmy mají 2 základní genetické operátory. Těmi jsou křížení a mutace. **Křížení** představuje kombinaci částí rodičů v nové jedince. Druhy křížení rozlišujeme podle počtu bodů, ve kterých rozdělíme chromozómy rodičů. Konkrétně známe jednobodové křížení, kde rozdělíme chromozómy rodičů v jednom bodě a druhé části chromozómů vyměníme. Další možnost je dvoubodové křížení. U této možnosti probíhá výměna střední části chromozómů mezi dvěma náhodně zvolenými body. Poslední možností je vícebodové křížení. Zde se vyměňují segmenty mezi zvoleným počtem bodů. Pro lepší představu viz Obr. 5.



Obr. 5: a) Schéma jednobodového křížení; b) Schéma dvoubodového křížení; c) Schéma vícebodového křížení [1]

Mutace je druhý genetický operátor klasických genetických algoritmů. Účelem tohoto operátoru je udržet diverzitu (rozmanitost) populace. Pokud by se provádělo pouze křížení, mohla by nastat situace, kdy by se křížili jedinci tolik, že by se přestali vytvářet noví jedinci, pouze by vznikali jedinci, kteří už v populaci figurovali. Mutace je v podstatě proces, kdy je každý gen (bit, parametr jedince) každého jedince s velmi malou pravděpodobností převrácen do opačné hodnoty.

2.4.5 Elitismus

Do genetických algoritmů byla také zavedena metoda elitismu, která zajišťuje přenos nejlepšího jedince z předchozí populace do nové. To zabraňuje mezigenerační ztrátě nejlepšího jedince.

2.4.6 Generační výměna

Po využití výše zmíněných operací dostáváme novou populaci jedinců. Tomuto kroku se souhrnně říká reprodukce. Důležitým předpokladem reprodukce je, aby nově vzniklá populace byla, díky vlivu selekce kvalitnějších rodičů a jejich křížení, lepší než ta předchozí. Touto novou populací pak nahrazujeme populaci předchozí. Takovýto cyklus se opakuje, dokud nenastane podmínka ukončení procesu.

2.4.7 Podmínka ukončení

Aby se tento cyklus neopakoval donekonečna, je zavedena podmínka ukončení. Nejčastější podmínky ukončení jsou [12]:

- dosažení maximálního počtu generací
- dosažení povoleného času běhu algoritmu
- nalezení uspokojivé hodnoty vhodnosti nejlepšího jedince
- nedostatečné zlepšování nejlepšího jedince během určitého počtu cyklů

2.5 Diferenciální evoluce

Diferenciální evoluce je jeden z nejnovějších evolučních algoritmů. Tento algoritmus byl vyvinut a poprvé využit K. Pricem a R. Stornem v USA. Schématem je velmi podobný genetickému algoritmu, ze kterého v podstatě vychází a s nímž má několik společných charakteristik. Zmiňme charakteristiky jako je například tvorba potomků, užití generací atd.

Stěžejním momentem pro diferenciální evoluci bylo objevení tzv. *diferenciální mutace*. Tato mutace spočívá v generaci zkušebního řešení pomocí přičtení diference dvou náhodně zvolených jedinců v populaci ke třetímu. Mutaci objevil K. Price při zkoumání a úpravě genetického žíhání. Ze spojení těchto dvou algoritmů vzešlo to, co je dnes nazýváno první verzí diferenciální evoluce. Algoritmus prošel přirozeným vývojem a v dnešní době je uznáván odbornou veřejností [13].

2.5.1 Stanovení parametrů

Ačkoliv se nepodílí na vlastní činnosti algoritmu, je tento krok stěžejní pro kvalitu diferenciální evoluce. Parametry diferenciální evoluce jsou: **F** – mutační konstanta, její hodnota se obvykle nastavuje v intervalu 0 až 2 a využívá se při reprodukčním cyklu, který bude vysvětlen později, **CR** – práh křížení, ten se využívá při vytváření tzv. *zkušebního vektoru*, volíme ho v rozmezí 0 až 1, **NP** – počet jedinců v populaci, nejčastěji

se používá desetinásobek rozměru jedince, **D** – rozměr jedince, nebo také počet argumentů účelové funkce. Rozměr jedince je volen s ohledem na řešený problém [13].

2.5.2 Tvorba populace

V diferenciální evoluci je s populací nakládáno stejně jako u jiných evolučních algoritmů. Podle zvoleného parametru **NP** vygenerujeme daný počet jedinců. Každý jedinec má pevnou délku **D**. Každý parametr jedince je generován náhodně ve zvoleném rozmezí, tak aby bylo řešení reprezentované jedincem smysluplné [13].

2.5.3 Reprodukční cyklus

V průběhu každé generace se provádí cyklus, který je u diferenciální evoluce nazván jako reprodukční. Úkolem tohoto cyklu je zaručení využití každého jedince pro reprodukci. Tento cyklus prochází jedince v populaci jednoho po druhém a provádí s nimi reprodukční operace popsané dále. Každá generace je ukončena po tom, co cyklus projde všechny jedince v populaci.

S každým zvoleným jedincem cyklus provádí mutaci a křížení. To probíhá za pomoci čtyř rodičů. Na rozdíl od genetického algoritmu, kde jsou využiti pouze dva rodiče. Jeden rodič je tedy aktuálně vybraný jedinec z populace a zbylí tři rodiče jsou náhodně vybráni ze zbylých jedinců v populaci. Je důležité zajistit, aby se vybraní jedinci neopakovali. Jako další krok nastává mutace. V tomto kroku je ze tří vybraných jedinců vytvářen tzv. *šumový* (někdy také mutační, zmutovaný) *vektor* ***V*** [13]. Tento vektor se vypočítá pomocí rovnice (3), kde X_{r1}, X_{r2}, X_{r3} jsou náhodně vybraní rodiče z populace a F je mutační konstanta.

$$V = X_{r1} + F \times (X_{r2} - X_{r3}) \tag{3}$$

Tato varianta výpočtu šumového vektoru se označuje anglickou zkratkou DE/rand/1. DE v této zkratce označuje diferenciální evoluci (anglicky Differential Evolution). Slovo za lomítkem je *rand*, to je zkratka pro *random* (česky náhodný), což je vektor, který je využit jako základ pro mutaci. V jiné variantě zde může stát slovíčko *best* (česky nejlepší), jako základ pro mutaci se potom nebere náhodně vybraný vektor, ale vždy nejlepší nalezený vektor (jedinec) aktuální populace. Číslo za druhým lomítkem označuje počet diferenčních (rozdílových) vektorů použitých při výpočtu. V tomto případě číslo 1 označuje 1 použitý diferenční vektor a to $X_{r2} - X_{r3}$. Existuje více způsobů pro výpočet šumového vektoru. Podle těchto způsobů rozlišujeme jednotlivé druhy diferenciální evoluce. Některé zajímavé varianty diferenciální evoluce jsou uvedeny v rovnicích (4) až (8) [14]:

$$\text{DE/rand/2:} V = X_i + F \times (X_{r1} - X_{r2}) + F \times (X_{r3} + X_{r4}) \tag{4}$$

$$\text{DE/best/1: } V = X_{best} + F \times (X_{r1} - X_{r2}) \tag{5}$$

$$\text{DE/best/2: } V = X_{best} + F \times (X_{r1} - X_{r2}) + F \times (X_{r3} - X_{r4}) \tag{6}$$

$$\text{DE/current-to-rand/1: } V = X_i + K \times (X_{r1} - X_i) + F' \times (X_{r2} - X_{r3}) \quad (7)$$

$$\text{DE/rand-to-best/1: } V = X_{r1} + F \times (X_{best} - X_{r2}) + F \times (X_{r3} - X_{r4}) \quad (8)$$

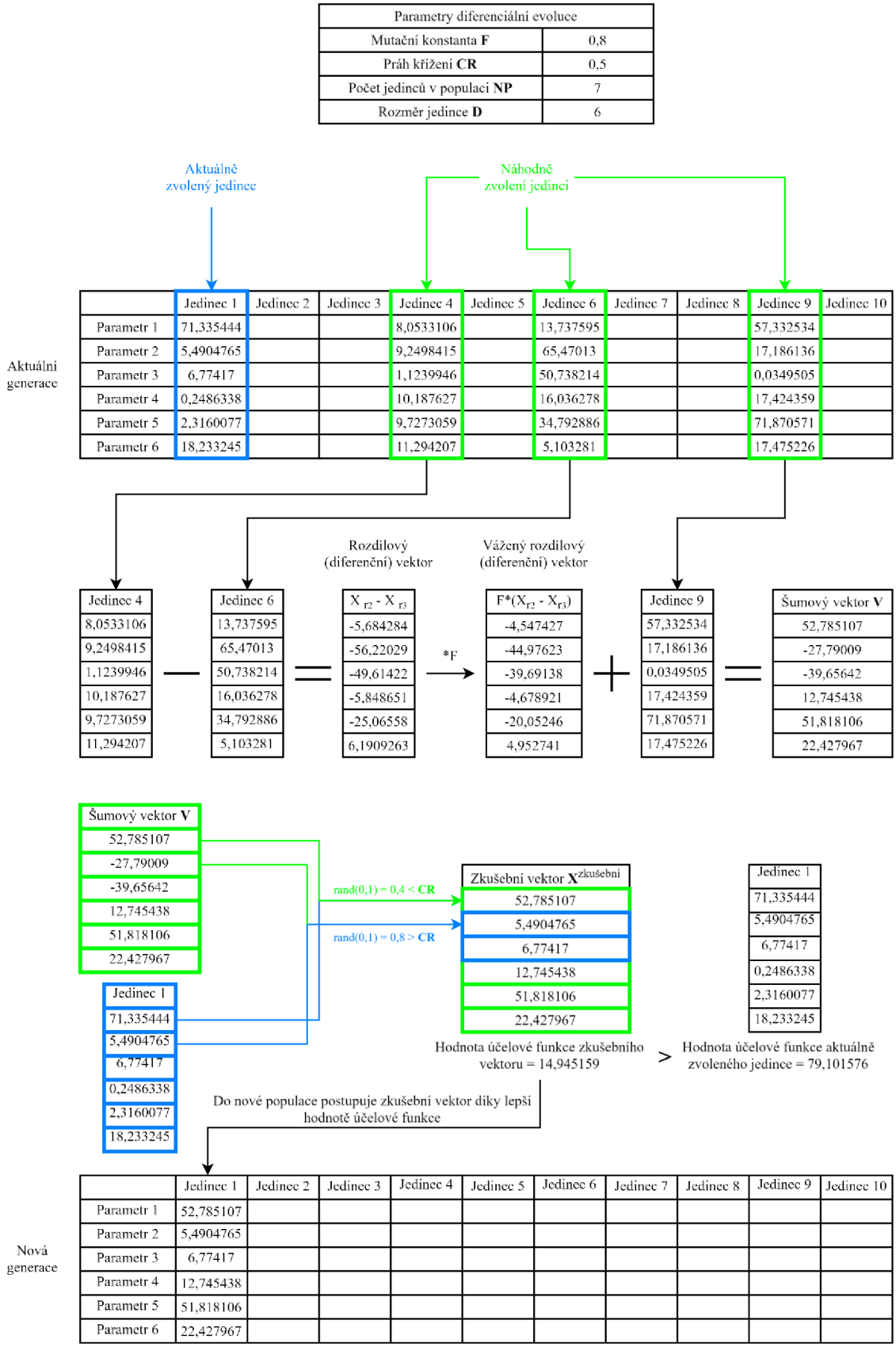
V rovnicích (4) až (8) označuje X_i aktuálně vybraného jedince z populace, pro kterého je vytvářen šumový vektor. Speciální koeficienty v rovnici (7) jsou: K, ten označuje kombinační koeficient, který by měl být zvolen v rozmezí od 0 do 1, dalším je F', který je vypočten z rovnice $F' = K \times F$.

Jakmile je vytvořen šumový vektor, nastává čas pro proces křížení. Cílem křížení je vytvoření potomka, tzv. *zkušebního vektoru*. Ten by měl obsahovat část šumového vektoru a jeho zbytek by měl být tvořen čtvrtým, doposud nevyužitým potomkem. Vytvářený zkušební vektor bude mít pevnou délku **D**. Jednotlivé parametry se vybírají následujícím způsobem: nejprve vybereme oba první parametry z šumového vektoru a čtvrtého jedince, pro tuto dvojici je vygenerováno náhodné číslo mezi 0 a 1, které je porovnáno s prahem křížení **CR**, pokud je vygenerované číslo menší než práh křížení, je vybrán parametr šumového jedince, v druhém případě je jako příslušný parametr zkušebního jedince vybrán parametr čtvrtého rodiče. Tento postup se opakuje pro každý parametr zkušebního vektoru. Tento postup je matematicky vyjádřen rovnicí (9).

$$x_j^{zkušební} = \begin{cases} V_j, jestliže\ rnd(0,1) \leq CR \\ X_{i,j}\ v\ ostatních\ případech, \end{cases} \quad (9)$$

kde $x_j^{zkušební}$ je j-tý parametr zkušebního vektoru, V_j je j-tý parametr šumového vektoru, $X_{i,j}$ je j-tý parametr aktuálně zvoleného jedince a $rnd(0,1)$ je pseudonáhodně vygenerované číslo v rozmezí 0 až 1.

V posledním kroku reprodukčního cyklu jsou porovnány hodnoty účelových funkcí aktuálního jedince a zkušebního vektoru. Do nové generace je zvolen vektor, který dosáhnul lepšího výsledku účelové funkce. Uvedený postup je opakován pro každého jedince v populaci. Pro lepší představu je tento cyklus znázorněn na Obrázku 6.



Obr. 6: Grafické znázornění reprodukčního cyklu diferenciální evoluce, překresleno z [13]

2.5.4 Kontrola podmínky ukončení algoritmu

Na konci každého reprodukčního cyklu, tedy vytvoření nové generace, nastává čas pro kontrolu ukončovací podmínky. Běžně diferenciální evoluce probíhá, dokud není naplněn uživatelem určený počet generací. Nikde však není zakázáno vytvoření vlastní ukončovací podmínky. Například je možné ukončit diferenciální evoluci, pokud se v posledních 10 generacích nezměnila hodnota účelové funkce nejlepšího jedince.

Jestliže podmínka ukončení není splněna, pokračuje diferenciální evoluce opět reprodukčním cyklem prováděném na nově získané populaci (generaci) a takto se proces opakuje, než je podmínka ukončení splněna [13].

3 APLIKACE EVT

EVT jsou všeobecně velmi zajímavým a hojně využívaným odvětvím optimalizace. Jak genetické algoritmy, tak diferenciální evoluce jsou využívány pro řešení velké řady inženýrských problémů. Jako příklad využití genetických algoritmů uveďme například optimalizační model pro redukci spotřeby energie při procesu čištění zemního plynu od síry nebo algoritmus pro nakládaní kontejnerů v přístavu. V obou zmíněných příkladech byl genetický algoritmus použit úspěšně. Pro diferenciální evoluci zmiňme jako příklad třeba rozvržení polohy větrných elektráren pro maximalizaci získané energie či optimalizaci spotřeby energie při skladování stlačeného vzduchu. Použití diferenciální evoluce se v těchto případech setkalo s pozitivními výsledky [5].

Z nepřeberného množství aplikací evolučních technik byl pro tuto práci zvolen problém hledání cesty mobilního robota. Daný problém spočívá v nalezení optimální trasy mobilního (pohyblivého) robota. Účelem je najít trasu takovou, aby byla co nejkratší a zároveň se vyhnula všem překážkám. Zvolený problém je detailněji popsán v následující podkapitole.

3.1 Problém navigace mobilního robota

V moderní průmyslu snažícím se o automatizaci začínají dostávat čím dál tím více prostoru roboti. Jejich využití přináší velký potenciál, ale zároveň s tím vyvstává také řada problémů jako například pohyb robotů, jejich přesouvání mezi pracovními stanovišti, optimální provedení úkonů, které musí robot vykonat atd. Jako zajímavé řešení těchto problémů se jeví implementování určité umělé inteligence do těchto robotů. Díky tomu jsou pak schopní se sami rozhodovat na základě informací, které mají k dispozici. Právě zde přicházejí na scénu již zmíněné EVT.

Konkrétně nacházejí uplatnění v navigaci mobilních robotů. Byť se to nemusí na první pohled zdát, je pro autonomního robota navigace velmi složitým problémem. Tento problém spočívá v nalezení optimální cesty mezi startovní a cílovou pozicí. Optimální cestu většinou představuje nejkratší bezkolizní cesta. V potaz se nicméně může brát i spotřeba energie robota či rychlost nalezení trasy.

Navigace představuje jeden ze základní problémů mobilních robotických systémů. Aby byl robot schopen dokončit úspěšně svůj úkol, musí znát jednak svou polohu v prostoru, jednak také znát pozici v prostoru, do které se má dostat. Navíc také musí brát v potaz nebezpečí nastávající při samotném pohybu. Toto nebezpečí představují překážky a jiné okolní vlivy. V jednoduchosti se problém navigace dá shrnout jako hledání odpovědí na 3 základní otázky: Kde jsem? Kam se potřebuji přesunout? Jak se tam dostanu? Tyto otázky zodpovídají 3 základní navigační funkce, a to jsou lokalizace, mapování a plánování cesty [15].

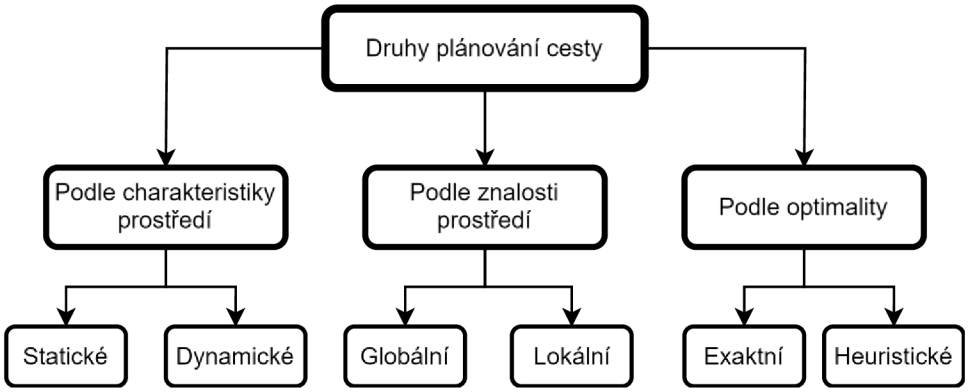
- *Lokalizace*. Úkolem této funkce je nalézt odpověď na otázku: Kde jsem? V současné době se pro lokalizace používá spousta různých metod. Ve

venkovních prostorech dá použít GPS. Další možnosti jsou kamery nebo jiné senzory jako například ultrazvukové nebo laserové.

- *Mapování.* Aby bylo možné zodpovědět otázku kam se potřebuji přesunout, je potřeba, aby měl robot povědomí o svém okolí. Na to se soustředí funkce mapování. Mapa umožňuje robotu rozlišovat směry a různé lokace, které už navštívil nebo do nich teprve směřuje. Mapu může představovat například graf či matice.
- *Plánování cesty.* Funkce plánovaní cesty zodpovídá za nalezení nejlepší cesty mezi počáteční a koncovou pozicí. Pracuje s předpokladem, že obě tyto pozice už jsou dopředu známé. Na aplikaci umělé inteligence pro řešení této funkce je soustředěna tato práce. Na funkci plánování cesty je aplikován genetický algoritmus a diferenciální evoluce.

3.2 Plánování cesty mobilního robota

Samotné plánování cesty je stále dosti složitým problémem. Je třeba zde brát ohled na jisté faktory a těmi jsou: charakteristika prostředí, úroveň znalosti prostředí a také optimalita použitého algoritmu. Toto rozdělení je graficky znázorněno na Obrázku 7.



Obr. 7: Druhy plánování cesty, překresleno z [15]

Podle charakteristiky prostředí dělíme algoritmy plánování cesty mobilního robota na algoritmy pro *statické prostředí* a algoritmy pro *dynamické prostředí*. Statické prostředí je, jak jeho název napovídá, statické tedy nepohyblivé. Startovní i cílový bod je v tomto prostředí pevně daný a neměnný. Taktéž překážky jsou statické, což znamená, že se v čase nemění ani jejich poloha ani tvar. Naopak u dynamického prostředí se může cílový bod nebo poloha překážek měnit v průběhu plánování trasy. Kvůli tomu je algoritmus plánující trasu v dynamickém prostředí nucen se přizpůsobovat a dynamicky reagovat na změny v prostředí v reálném čase. Algoritmy plánující trasu v dynamickém prostředí jsou tedy obecně složitější a náročnější než algoritmy pro statické prostředí.

Pro plánování trasy je důležitá informace o okolí robota. Tuto informaci představuje mapa definující současnou polohu robota, cíl a polohu překážek. Podle toho, kolik informací robot o svém okolí má, dělíme plánování cesty na dvě stěžejní oblasti. První

oblastí je *globální plánování cesty*. Tato oblast předpokládá, že robot má zmapované své okolí. Naopak druhá oblast předpokládá, že robot nezná své okolí a spoléhá na detekování překážek pomocí senzorů a sestavuje mapu svého okolí v reálném čase. Druhá oblast se nazývá *lokální plánování cesty*.

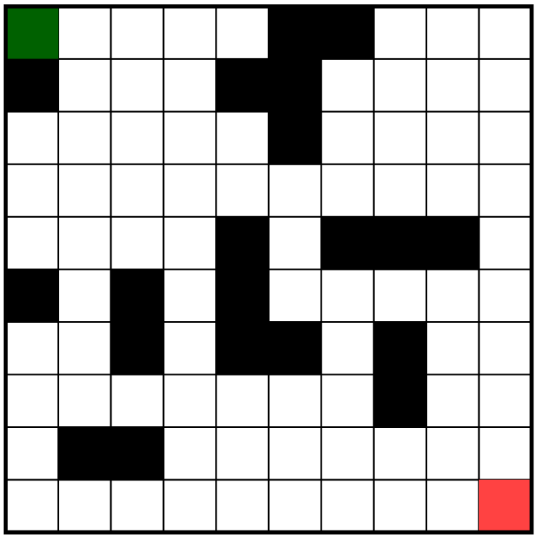
Poslední druh dělení algoritmů pro plánování trasy se nevztahuje k prostředí ale k optimalitě algoritmu (optimální algoritmus je schopen najít optimální řešení, pokud takové řešení existuje). Podle toho můžeme algoritmy dělit na *exaktní* a *heuristické*. Použití exaktního algoritmu spočívá v nalezení optimálního řešení, případně dokázání, že takové řešení neexistuje. Heuristické algoritmy nehledají přesně optimální řešení. Výstup heuristického algoritmu je řešení pouze přibližné optimálnímu. Logicky tedy není tolik přesné jako od exaktního algoritmu. Výhodou naopak je, že toto přibližné řešení je získáno ve znatelně kratším čase [15].

3.3 Reprezentace prostředí

Dalším zásadním faktorem pro algoritmus hledání trasy je prostředí, ve kterém bude pracovat. Reálné prostředí je velmi složité, a tak se reprezentuje zjednodušeně. V případě této práce je prostředí, ve kterém bude hledána trasa, zjednodušeno do 2D (dvourozměrné mapy). Prostředí může být reprezentováno dvěma základními typy: *diskrétní prostředí* nebo *spojité prostředí*. Volba reprezentace prostředí má zásadní vliv na kvalitu a robustnost algoritmu [16].

Diskrétní prostředí

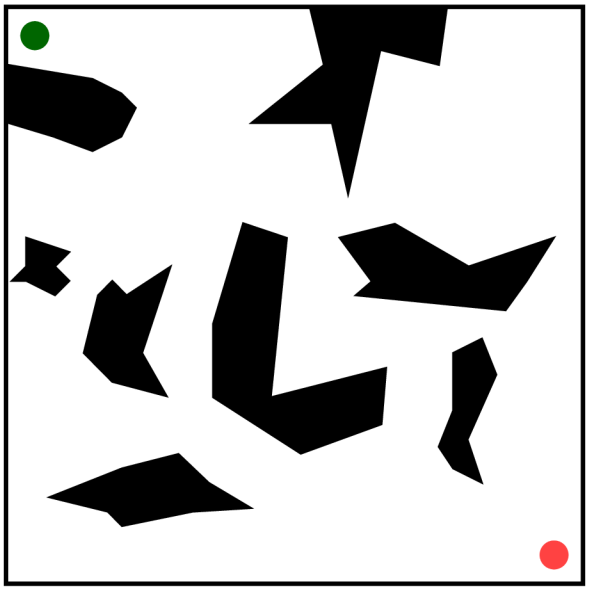
Tento druh reprezentace prostředí spočívá ve zjednodušení reálného světa pomocí buněk či uzlů a hran grafu viditelnosti. Taková reprezentace může vypadat například jako na obrázku 8. Hlavní nevýhoda buňkové reprezentace spočívá v její přesnosti. Konkrétně je reprezentace překážek omezena právě na velikost buněk. I když překážka zasahuje do buňky pouze malou částí, buňka je stejně označena celá jako překážka. Samozřejmě se dají pro zvětšení přesnosti zmenšit buňky. To ale přináší zvyšující se výpočetní náročnost. Další nevýhodou je omezení pohybu robota na jednotlivé buňky. Je tedy nutné přijmout kompromis mezi přesností a výpočetní náročností.



Obr. 8: Diskrétní prostředí reprezentované pomocí buněk

Spojité prostředí

Spojité prostředí se ve své reprezentaci více přibližuje reálnému prostředí. Na rozdíl od diskrétní reprezentace už nejsou překážky ani pohyb robota omezeny na jednotlivé buňky. Zvýšená přesnost opět přináší větší výpočetní náročnost. Příklad spojité reprezentace prostředí je na obrázku 9.



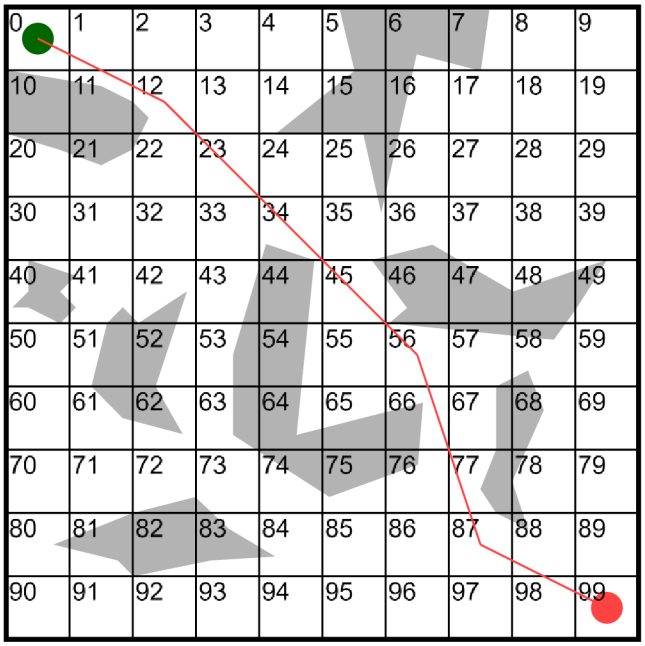
Obr. 9: Spojité prostředí

4 IMPLEMENTACE

Samotná vlastní aplikace genetického algoritmu a diferenciální evoluce se soustředí na heuristické globální plánovaní trasy ve statickém prostředí. Jsou tedy uvažovány následující předpoklady: startovní a cílová poloha robota jsou předem známé a pevně dané, překážky jsou reprezentovány konvexními mnohoúhelníky, jejichž souřadnice jsou předem známé a neměnné.

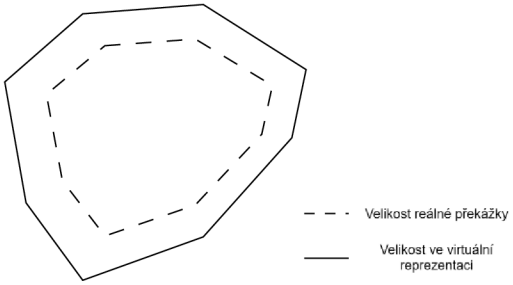
Virtuální prostředí pro implementaci výše zmíněných algoritmů bylo vytvořeno za pomoci programovacího jazyka *Python*. Tento programovací jazyk byl upřednostněn před jinými, třebaže rychlejšími (myšleno tak, že implementace v jiném jazyce by mohla pracovat rychleji), zejména kvůli jeho přehlednosti, relativní jednoduchosti a čitelnosti. Python je open source programovací jazyk, který obsahuje dynamickou kontrolu datových typů a zároveň podporuje moderní programovací paradigmata, jako například při implementaci využitého objektově orientovaného programování. Další výhoda Pythonu spočívá v dostupnosti a pestrosti různých knihoven a přídavných modulů. Jeden z těchto modulů *Pygame*, byl využit pro vizualizaci virtuálního prostředí. Tento modul přináší knihovny pro práci s grafikou, zvukem a vstupními zařízeními [16]. Poslední použitou knihovnou je *NumPy*. Tato knihovna pro Python přidává podporu práce s vícerozměrnými poli, maticemi a také spoustu dalších matematických funkcí. Konkrétně jsou v práci využity funkce pseudo-náhodného generování čísel a funkce pro práce s poli hodnot [17].

Samotné prostředí je reprezentováno kombinací spojitého a diskrétního modelu. Prohledávaný prostor je rozdělen na mřížku o velikosti 50x50 buněk. Pro představu je na obrázku 10 vyobrazena podobná mřížka o velikosti 10x10 buněk. Jednotlivé buňky jsou očíslovány. Tyto buňky však mají funkci pouze jako záchytné body pro plánování cesty a nijak neomezují překážky. Ty jsou reprezentovány jako ve spojitém modelu. Jsou tedy definovány pomocí svých hran a vrcholů. Překážky jsou ovšem omezeny na konvexní mnohoúhelníky z důvodu výpočetní složitosti. Toto řešení bylo inspirováno prací [18].



Obr. 10: Diskrétní prostředí s očíslovanými buňkami, spojitými překážkami a příkladem potenciálního řešení nalezeného algoritmem (červená trasa)

Poslední důležitý předpoklad týkající se reprezentace prostředí je rozměr překážek. Ten je zvětšen o bezpečnou vzdálenost (viz obrázek 11). Bezpečná vzdálenost se vypočítá z rozměrů robota, ke které se přičte menší rezerva. Díky tomuto může být robot reprezentován bodem. Pokud tedy dojde k doteku nalezené trasy s překážkou, tak by měla tato bezpečná vzdálenost zajistit, že ke kolizi nedojde.



Obr. 11: Překážka ve virtuální reprezentaci zvětšená o bezpečnou vzdálenost

4.1 Účelová funkce

Definice účelové funkce je zásadním krokem pro aplikace obou algoritmů. Pro účely této práce byla pro obě aplikace zvolena stejná účelová (cenová) funkce. Je definována pomocí rovnice (10).

$$f_{cost} = d + k \times p, \tag{10}$$

kde f_{cost} představuje právě hodnotu účelové funkce, d označuje délku trasy, p je hodnota penalizace a k je koeficient penalizace.

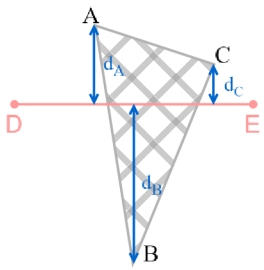
Zde zvolená funkce tedy bere v potaz délku trasy a také míru průniku s překážkami. Tuto funkci se snažíme minimalizovat, což znamená, že se snažíme, aby byla nalezená cesta co nejkratší. Dále se snažíme o nulovou penalizaci. Jak se tato penalizace počítá, bude vysvětleno v dalším textu. Koeficient penalizace udává její váhu, tj. jak moc je penalizace důležitá.

Penalizace

Penalizace udává danému řešení postih za protínání překážek. Konkrétně udává nejnižší možnou vzdálenost, o kterou je potřeba cestu posunout, aby už překážku neprotínala. Vzdálenost se počítá jako délka úsečky kolmé na trasu. Počítáme ji pro každý úsek nalezené trasy (potencionálního řešení) zvlášť a následně je všechny sečteme, viz rovnice (11).

$$p = \sum_{i=1}^{N} a_i, \tag{11}$$

kde N je počet úseků trasy a a_i je penalizace i-tého úseku.

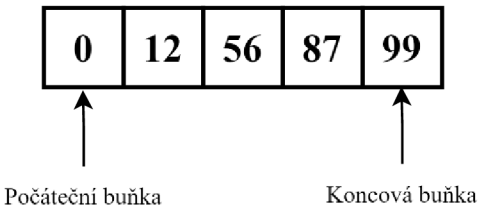


Obr. 12: Výpočet penalizace a_i pro úsek trasy $|DE|$

Na obrázku 12 můžeme vidět úsek trasy $|DE|$, který prochází překážkou reprezentovanou trojúhelníkem ABC. Nejdříve jsou vypočteny vzdálenosti přímky k jednotlivým vrcholům d_A, d_B, d_C. Vzdálenost d_C je sice nejkratší, ale bohužel není dostatečná k posunutí úseku z překážky. Jako penalizace je tedy zvolena vzdálenost d_A.

4.2 Reprezentace jedince

V obou aplikacích je jedinec reprezentován jako řetězec čísel. Každé číslo v řetězci označuje buňku ve virtuálním prostředí. Tato reprezentace umožňuje jednoduchou práci s jednotlivými jedinci jak genetickému algoritmu, tak diferenciální evoluci. Příklad takového jedince je uveden na obrázku 13. Na obrázku 10 můžeme vidět, jak daný jedinec vypadá ve virtuální reprezentaci.

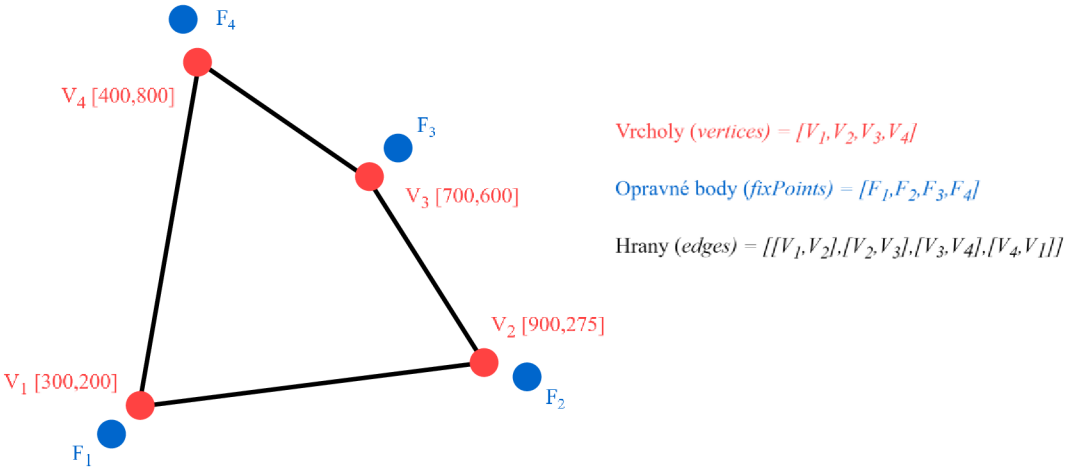


Obr. 13: Reprezentace jedince

Jako první číslo v řetězci je vždy uvedeno číslo počáteční buňky, které je předem známo. Stejně tak na posledním místě je číslo buňky cíle. Mezi nimi se nachází jednotlivé buňky, kterými trasa prochází. Důležitou vlastností jedince je jeho délka. V případě genetického algoritmu není tato délka pevně stanovená, což se později ukáže jako velká výhoda [18]. Diferenciální evoluce však umožnuje práci pouze s jedinci pevně definované délky.

4.3 Reprezentace překážek

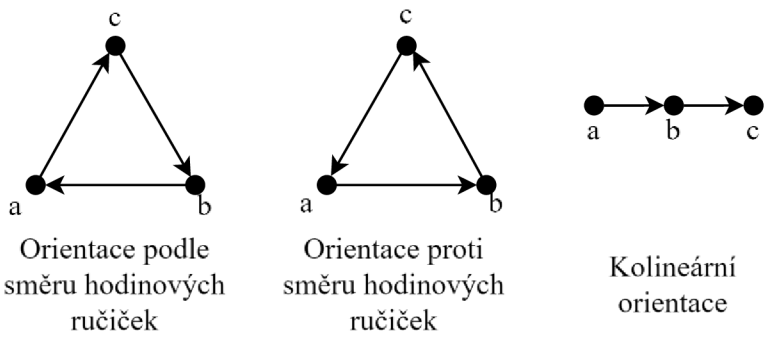
V souladu s metodikou objektově orientovaného programovaní je v této práci pro reprezentaci překážek vytvořena třída *Obstacles* (překážky) zastupující všechny překážky jako celek a její podtřída *Obstacle*. Každá překážka je definovaná pomocí pole bodů, které představují její vrcholy. Z těchto bodů můžeme jednoduše vytvořit hrany každé překážky, což bude využito při detekci kolize. Dále jsou pomocí bodů vytvořeny tzv. *fixPoints* (opravné body) každé překážky. Jsou to body blízké vrcholům posunuté mimo překážku směrem od jejího těžiště, které jsou využity při opravě jednotlivých úseků cesty pomocí genetického operátoru Oprava (viz podkapitola 4.5). Pro lepší představu je reprezentace překážky na obrázku 14. Čísla v hranatých závorkách představují souřadnice bodů.



Obr. 14: Reprezentace překážky

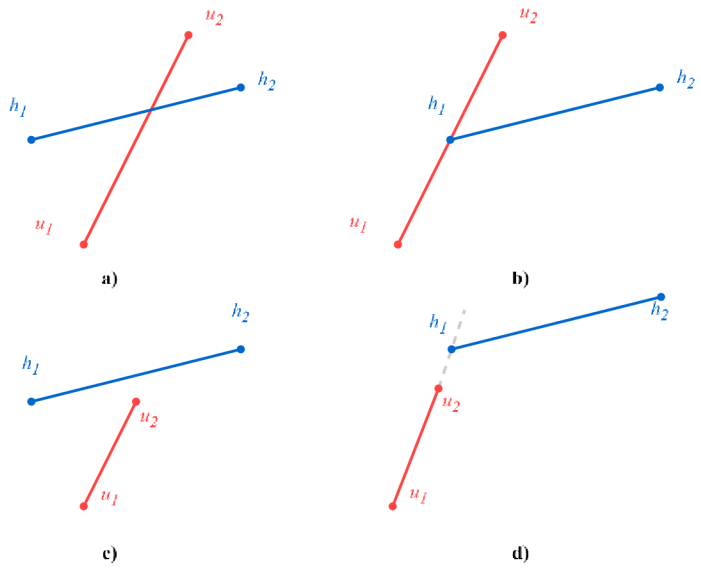
4.4 Detekce kolize

Detekce kolize je stěžejní problém hledání cesty mobilního robota. Aby bylo možné určit penalizaci nebo opravit případnou kolizní cestu, je nutné detekovat kolize. V této práci je důležité detekovat, na jaké části cesty dochází ke kolizi a také se kterou překážkou. Detekce kolize je vyřešena následujícím způsobem. Na každém jedinci probíhá kontrola po jednotlivých úsečkách. Tedy nejprve se kontroluje úsečka mezi prvním a druhým bodem a následně mezi druhým a třetím bodem a tak dále. U každé překážky známe hrany, kterými je tvořena. Tato situace vede na problém hledání průsečíků dvou úseček. V práci je problém řešen pomocí orientace jednotlivých bodů úseček. Na obrázku 15 jsou uvedeny možné druhy orientace bodů (a, b, c).



Obr. 15: Druhy orientace bodů (a, b, c)

Dvě úsečky (u_1, u_2) a (h_1, h_2) mají průsečík, právě když trojice bodů (u_1, u_2, h_1), (u_1, u_2, h_2) mají rozdílnou orientaci a trojice bodů (h_1, h_2, u_1), (h_1, h_2, u_2) mají také rozdílnou orientaci.



Obr. 16: a), b) příklady dvou úseček s průnikem; c), d) příklady dvou úseček bez průniku, překresleno z [19]

Na obrázku 16 můžeme vidět čtyři příklady vzájemné polohy dvou úseček. V příkladu a) je trojice bodů (u_1, u_2, h_1) orientovaná proti směru hodinových ručiček a trojice (u_1, u_2, h_2) je orientovaná po směru hodinových ručiček. První dvě trojice bodů tedy mají opačnou orientaci. Trojice bodů (h_1, h_2, u_1) je orientovaná po směru hodinových ručiček a trojice (h_1, h_2, u_2) je orientovaná proti směru hodinových ručiček. Druhé dvě trojice bodů mají taktéž rozdílnou orientaci, což potvrzuje, že obě úsečky mají průsečík. Opačný případ je v příkladu c). Zde můžeme vidět, že první dvě trojice sice mají opačnou orientaci nicméně trojice (h_1, h_2, u_1) je orientovaná po směru hodinových ručiček stejně jako trojice (h_1, h_2, u_2). Druhé dvě trojice mají stejnou orientaci, což dokazuje, že úsečky nemají průsečík [20].

4.5 Aplikace genetického algoritmu na problém hledání cesty mobilního robota

V prvním kroku aplikace genetického algoritmu je vygenerována počáteční populace. Jak bylo výše zmíněno, genetický algoritmus není omezený pevnou délkou jedince. V rámci tohoto kroku je nejprve vygenerováno náhodné celé číslo. Generování probíhá pomocí funkce *random.randint(2, maxPathLen)* knihovny *NumPy* [17]. V závorce jsou parametry funkce. Číslo 2 představuje minimální délku jedince. V našem případě takový jedinec tedy reprezentuje přímou trasu z počátku do cíle. Parametr *maxPathLen* představuje maximální délku generovaného jedince. Zde je zvolena maximální délka generovaného jedince 15. Poté je přistoupeno k sestavení řetězce. Jako první je vždy přidáno do řetězce číslo počáteční buňky. Následně je vygenerován příslušný počet mezibuněk podle požadované délky řetězce. Mezibuňka se generuje stejnou funkcí jako délka, pouze parametry jsou voleny podle počtu buněk ve virtuální reprezentaci. Jako poslední je přidáno číslo koncové buňky. Pomocí tohoto postupu je vygenerován jeden jedinec. Tento postup je opakován podle zvoleného počtu jedinců v populaci. V případě této práce je počet jedinců v populaci 20.

Poté je počáteční populace ohodnocena. Ohodnocení spočívá ve vypočítání hodnoty cenové funkce pro všechny jedince populace, následném spočítání vhodnosti a rozdělení pravděpodobností pro křížení. Cenová funkce již byla uvedena výše. V dalším kroku je hodnota cenové funkce každého jedince převedena na vhodnost jedince podle rovnice (12) převzaté z [1]. Hodnota vhodnosti se pohybuje mezi hodnotami 0 a 1. Z hodnoty vhodnosti jedince se potom podle rovnice (13) spočítá pravděpodobnost jeho vybrání pro rodičovství.

$$F(ind) = \frac{1}{f_{min} - f_{max}} \times [(1 - 0{,}01) \times f(ind) + f_{min} \times 0{,}01 - f_{max}], \qquad (12)$$

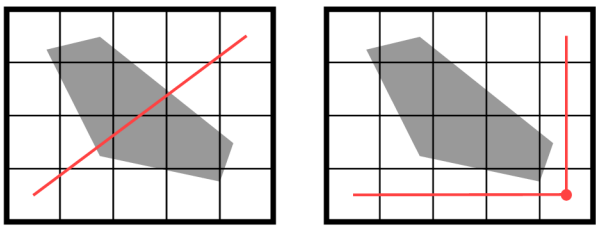
kde $F(ind)$ je vhodnost aktivního jedince, f_{min} je minimální hodnota účelové funkce, f_{max} je maximální hodnota účelové funkce, $f(ind)$ je hodnota účelové funkce aktivního jedince.

$$P(ind) = \frac{F(ind)}{\sum_{i=1}^{N} F(i)}, \tag{13}$$

kde $P(ind)$ je pravděpodobnost vybrání aktivního jedince pro rodičovství, $\sum_{i=1}^{N} F(i)$ je suma vhodností všech jedinců v aktuální populaci.

Jakmile je počáteční populace ohodnocena, začíná proces vytváření nové generace. Nejprve je použita metoda elitismu vysvětlená v rešeršní části. Nejlepší jedinec minulé generace se stává prvním jedincem nové generace. Dále přichází na řadu genetické operátory. V této práci je práci využito 5 genetických operátorů. Dva jsou standartní genetické operátory. Zbylé tři operátory představují vylepšení genetických algoritmů tak, aby byly vhodné pro hledání cesty mobilního robota. Vylepšující operátory jsou inspirovány prací [18]. Konkrétně jsou to operátory:

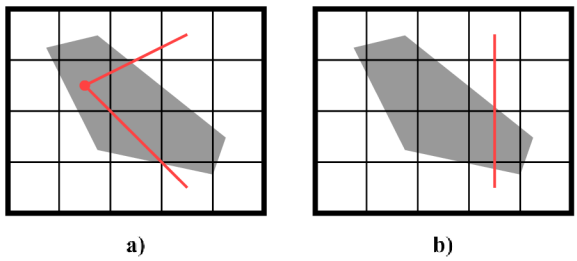
- **Křížení.** V aplikaci je využito jednobodové křížení popsané v rešeršní části s jedním rozdílem. Pro každého rodiče je zvolen zvlášť bod křížení. To je zapříčiněno potenciálně rozdílnou délkou rodičů. Rodiče pro křížení jsou vybírání z předešlé populace proporcionálním způsobem nebo taky ruletovým systémem.
- **Mutace.** Dále je nová generace zmutována. Mutační operátor prochází všechny buňky jedince s výjimkou první a poslední buňky, aby nebyly zmutovány počáteční a koncová buňka jedince. U každé buňky je vygenerováno náhodné číslo v rozmezí 0 až 1. Pokud je toto číslo menší než mutační konstanta, je na místo původní buňky vygenerováno náhodné číslo nové buňky. Mutační operátor takto prochází celou populaci, vynechává pouze prvního jedince, aby nedošlo ke ztrátě nejlepšího jedince v populaci.
- **Oprava**. Prvním vylepšujícím operátorem je oprava. Tento operátor prochází celého jedince po úsecích. Tedy úsek mezi prvním a druhým bodem, poté úsek mezi druhým a třetím a tak dále. Pokud úsek protíná překážku, operátor vkládá do řetězce nové číslo buňky tak, aby byl kolizní úsek rozdělen způsobem obcházejícím překážku (viz obrázek 17) [18].



Obr. 17: a) původní trasa; b) trasa opravená operátorem oprava

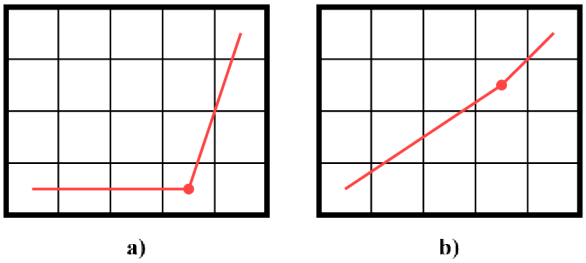
- **Vymazání.** Genetický operátor vymazání zlepšuje trasu následujícím způsobem. Operátor prochází jedince po trojicích bodů. V prvním kroku tedy bere v potaz první, druhý a třetí bod trasy. Následně zkusí operátor vymazat prostřední bod z trojice. Pokud dojde po vymazání bodu ke zlepšení hodnoty cenové funkce, je

bod vymazán natrvalo. Tímto způsobem prochází celého jedince. Příklad použití operátoru je na obrázku 18 [18].



Obr. 18: a) původní trasa; b) cesta opravená operátorem vymazání

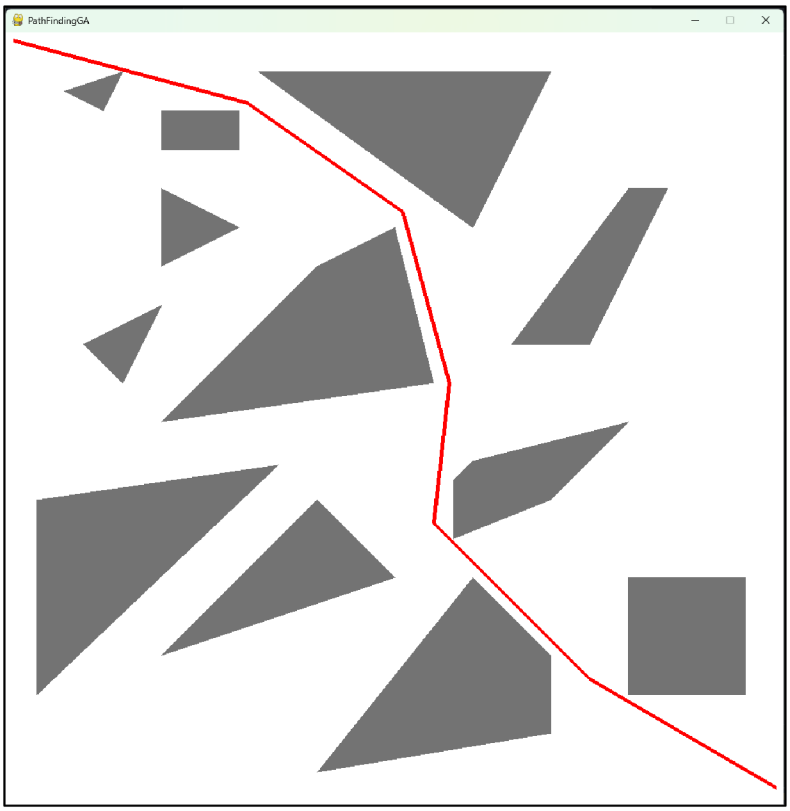
- **Zlepšení.** Posledním vylepšujícím operátorem je zlepšení. Ten prochází jedince jeden bod po druhém a aplikuje lokální prohledávání. Zkouší posouvat právě kontrolovaný bod o dvě buňky ve čtyřech směrech: nahoru, dolů, doprava a doleva. Do jedince je zvolen bod, pro který je hodnota cenové funkce nejlepší (viz obrázek 19). Tento operátor je určen k finálnímu doladění a vyhlazení cesty [18].



Obr. 19: a) původní trasa; b) trasa vylepšená operátorem zlepšení

Nová generace, ve které je zatím nejlepší jedinec minulé generace, je doplněna křížením na požadovaný počet. Dále přichází na řadu genetický operátor mutace. Když má nová generace zvolený počet jedinců a byla provedena mutace, přichází čas na zbylé tři operátory. Tyto operátory jsou relativně výpočetně náročné, a tak nemohou být volány všechny na každého jedince. Jsou tedy volány podobně jako mutace. Každý operátor prochází všechny jedince v populaci a s malou pravděpodobností jsou na právě aktivního jedince použity. Tuto pravděpodobnost určuje parametr mutační konstanty.

Poté je stará generace nahrazená nově vytvořenou. Na tuto generaci je znovu aplikován popsaný postup. Cyklus se opakuje, dokud není splněna podmínka ukončení. Zde je podmínka ukončení vyřešena pomocí proměnné *stopCount*, která začíná na nule. K proměnné se přičte jednička pokaždé, když je nejlepší jedinec nové generace stejný jako nejlepší jedinec minulé generace. Jakmile proměnná dosáhne hodnoty tři, tzn. že tři generace po sobě nedošlo ke zlepšení nejlepšího jedince, je algoritmus ukončen. Minimální počet generací je nastaven na pět. Před ukončením algoritmu jsou ještě na nejlepšího jedince použity operátory opravy, vymazaní a zlepšení. Výsledek testovacího běhu je na obrázku 20.

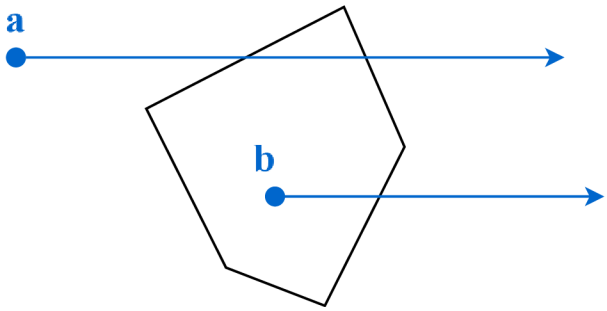


Obr. 20: Nejlepší jedinec testovacího běhu aplikace genetického algoritmu na hledání cesty

4.6 Aplikace diferenciální evoluce na problém hledání cesty mobilního robota

Při aplikaci diferenciální evoluce je opět potřeba nejdříve inicializovat počáteční populaci. Podobně jako u genetického algoritmu se zde zapojuje náhodné generování čísla buňky virtuálního prostředí. Opět je vytvářen zvolený počet jedinců tak, aby byl naplněn zvolený počet jedinců v populaci. Vytvoření každého jedince je provedeno následujícím způsobem.

Jako první číslo buňky je do jedince vždy přidáno číslo zvolené počáteční buňky. Následně je opět pomocí stejné funkce, jaká je použita u genetického algoritmu, vygenerováno číslo buňky, kterou bude cesta procházet. Náhodně vygenerované číslo buňky je zde kontrolováno, jestli se nenachází v překážce. Každé vygenerované číslo buňky se kontroluje vůči všem překážkám. Z vygenerovaného bodu (střed buňky) se vede vodorovná přímka do nekonečna ve směru osy x (viz obrázek 21). Spočítá se, kolikrát přímka protne hrany překážky. Bod se nachází uvnitř překážky, pokud je počet protnutí lichý nebo se bod nachází na hraně překážky. Příklad můžeme vidět na obrázku 21. Přímka bodu *a* protíná překážku 2krát a leží vně překážky. Přímka bodu *b* protíná překážku 1krát, je tedy uvnitř překážky. V takovéto situaci je vygenerováno nové číslo buňky. To se opakuje, dokud není vygenerován bod mimo překážku [21].

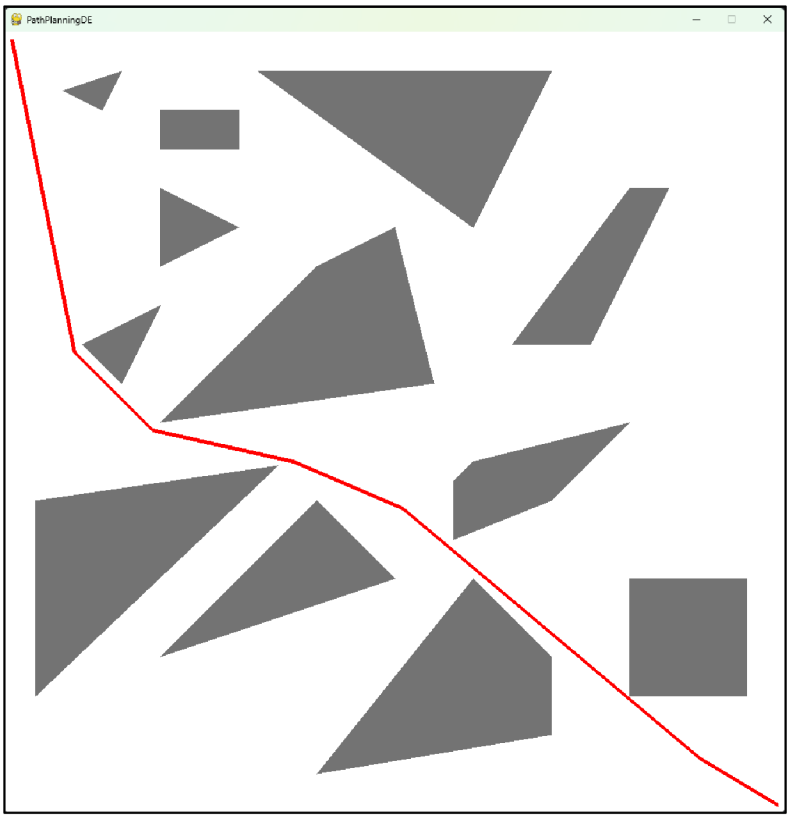


Obr. 21: Kontrola bodu ležícího uvnitř překážky [21]

Jako poslední bod je opět vložen požadovaný koncový bod. Tímto způsobem je naplněna celá populace.

Jakmile je naplněna počáteční populace, přichází čas pro mutaci, křížení a selekci. Mutace je prováděna tak, jak je popsáno v rešerši. Konkrétně je v práci využita varianta *DE/rand/1*. Mutační vektor je tedy vytvořen podle rovnice (3). V rámci vytváření mutačního vektoru jsou opět kontrolována jednotlivá čísla buněk, jestli se nachází v požadovaném rozsahu a jestli se nenachází v překážce. Pokud jsou mimo rozmezí nebo uvnitř překážky, je náhodně vygenerováno nové číslo buňky. Křížení je provedeno standartním způsobem popsaným v rešeršní části. Krok selekce je posledním krokem v každé generaci. Jak je popsáno v podkapitole 2.5.3, smysl selekce spočívá v rozhodnutí, zda je do nové generace předán starý jedinec nebo nově navrhnutý zkušební jedinec. Rozhoduje se podle hodnoty cenové funkce.

Tento proces je opakován podle počtu generací určeného uživatelem. Jako výsledek je pak předán nejlepší jedinec poslední populace. Bohužel zde není zaručeno, že tento jedinec nebude obsahovat žádné kolize. Za účelem odstranění kolizí a dohlazení nalezené cesty jsou tedy použity tři vylepšující operátory z genetického algoritmu popsané v podkapitole 4.5. Konkrétně jsou to tedy operátory: oprava, vymazání a zlepšení. Výsledek testovacího běhu je na obrázku 22.



Obr. 22: Nejlepší jedinec testovacího běhu aplikace diferenciální evoluce na hledání cesty

5 POROVNÁNÍ ALGORITMŮ

Pro porovnání obou algoritmů byla vygenerována dvě náhodná prostředí s různým počtem překážek. Každý algoritmus byl spuštěn 20krát v každém prostředí. Výsledky byly vypsány do tabulek a porovnány. Základní parametry obou algoritmů jsou uvedeny v následující tabulkách 1 a 2.

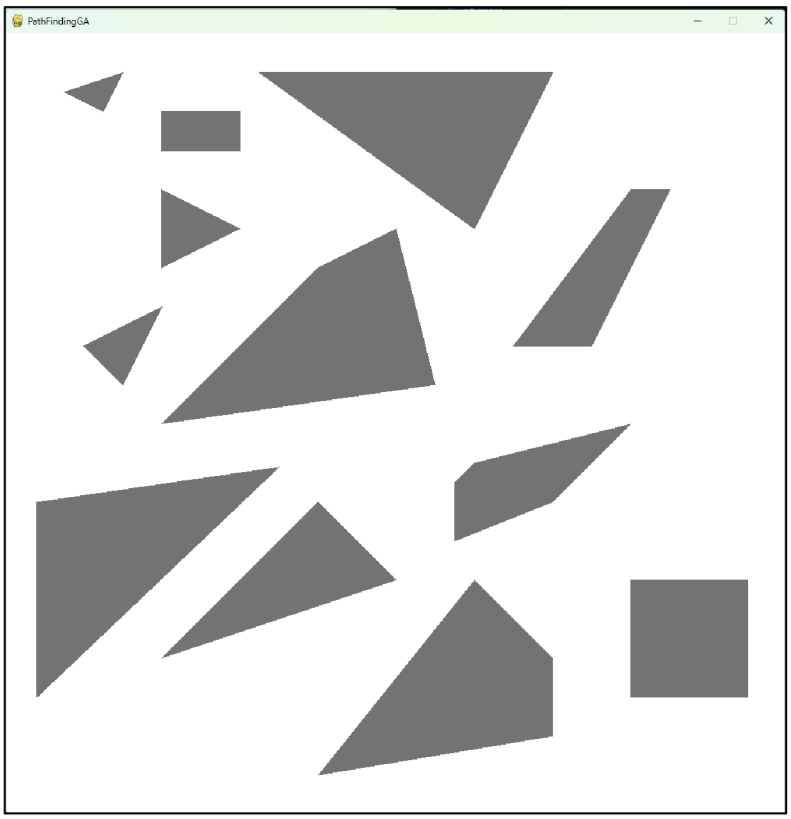
Parametr	Hodnota
Maximální délka jedince	15
Velikost populace	20
Mutační konstanta	0.1

Tab. 1: Spouštěcí parametry genetického algoritmu

Parametr	Hodnota
Mutační konstanta	0.2
Práh křížení	0.5
Délka jedince	6
Velikost populace	50
Počet generací	25

Tab. 2: Spouštěcí parametry diferenciální evoluce

První prostředí

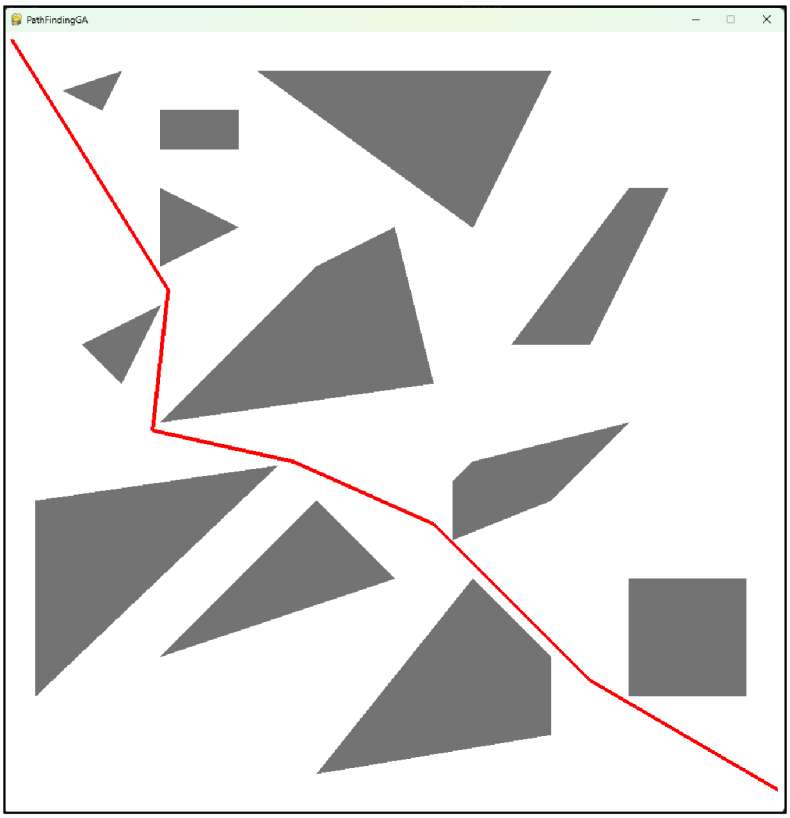


Obr. 23: Grafické zobrazení prvního prostředí

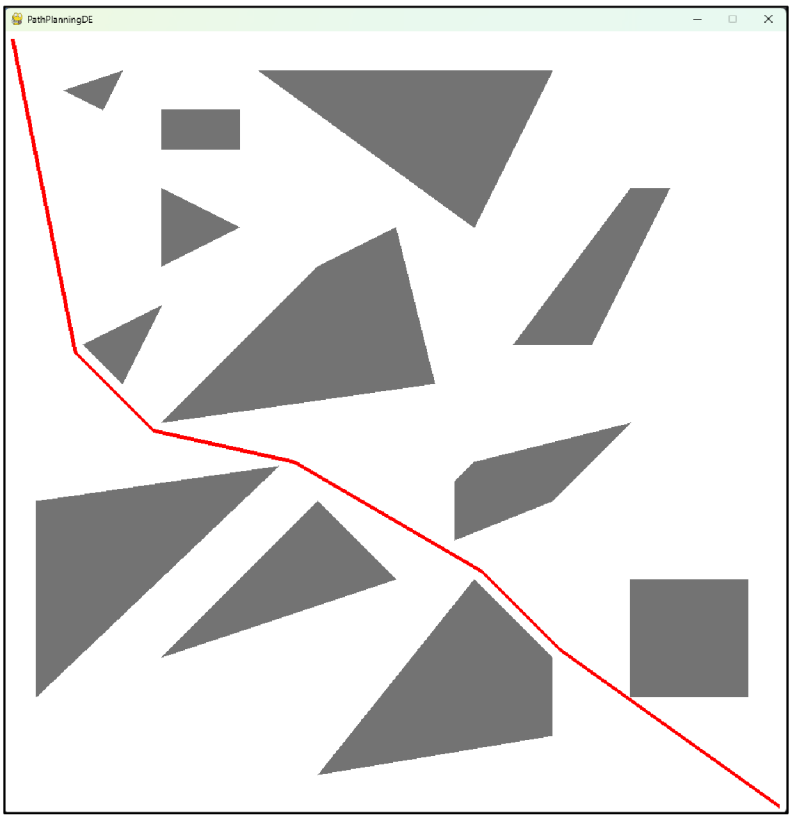
Algoritmus	Genetický algoritmus	Diferenciální evoluce
Průměrná délka nalezené trasy	1565.5	1512.1
Průměrný čas hledání řešení [s]	0.583	2.397
Délka nejlepší nalezené trasy	1501	1497
Délka nejhorší nalezené trasy	1834	1557

Tab. 3: Výsledky algoritmů v prvním prostředí

Na obrázku 23 je zobrazeno první prostředí, ve kterém byly algoritmy testovány. Hned za obrázkem v tabulce 3 jsou výsledky 20 běhů každého algoritmu. Výsledky říkají, že diferenciální evoluce dokázala nalézat kratší trasy. Genetický algoritmus naopak pracoval o dost rychleji. Zajímavostí je nejhorší nalezená trasa genetického algoritmu, v tomto případě můžeme vidět, že genetický algoritmus uváznul v lokálním minimu. Pro porovnání jsou nejlepší nalezené trasy zobrazeny na obrázcích 24 a 25.

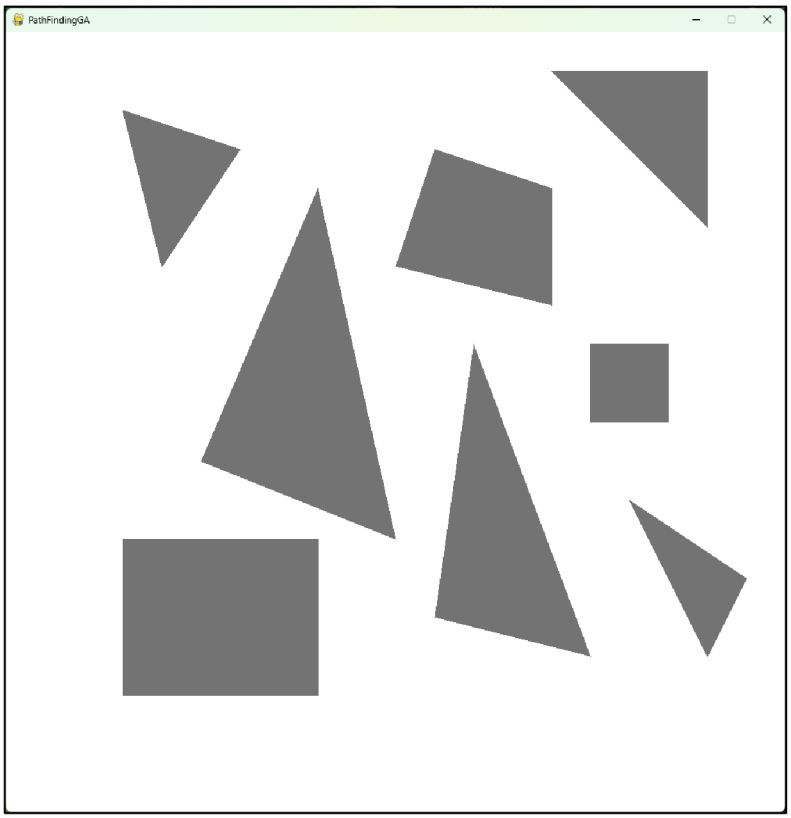


Obr. 24: Nejlepší trasa nalezená genetickým algoritmem v prvním testovacím prostředí



Obr. 25: Nejlepší trasa nalezená diferenciální evolucí v prvním testovacím prostředí

Druhé prostředí

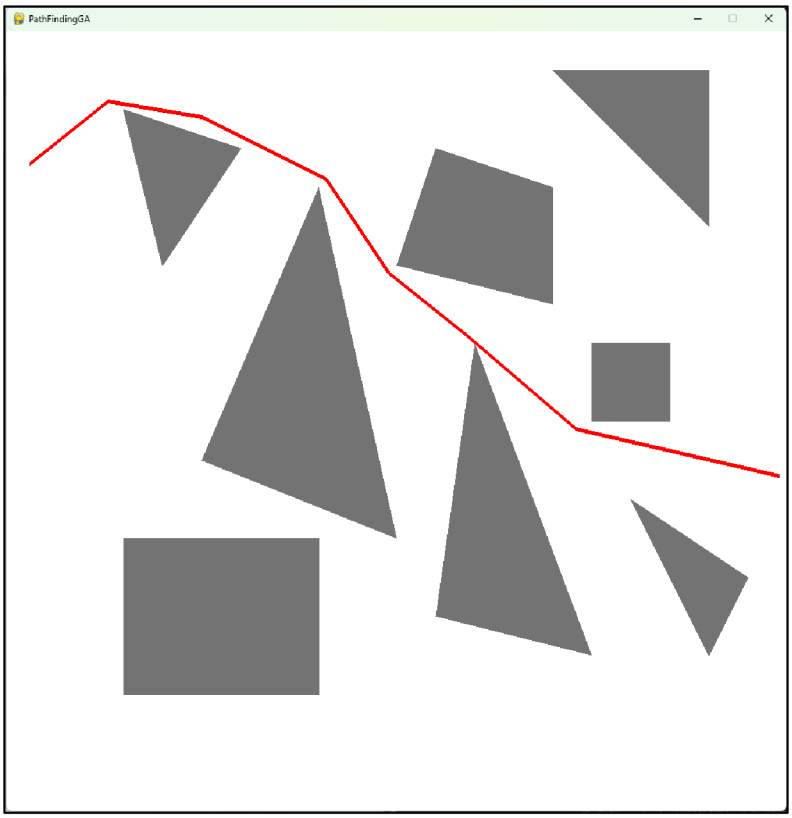


Obr. 26: Grafické zobrazení druhého prostředí

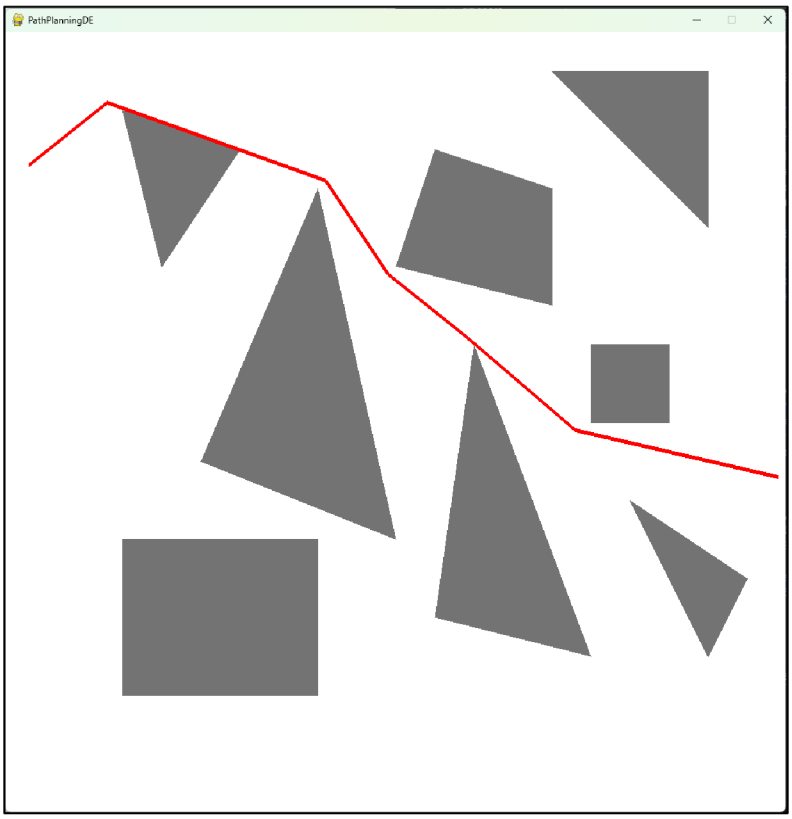
Algoritmus	Genetický algoritmus	Diferenciální evoluce
Průměrná délka nalezené trasy	1184.15	1215.4
Průměrný čas hledání řešení [s]	0.4645	1.5315
Délka nejlepší nalezené trasy	1152	1149
Délka nejhorší nalezené trasy	1241	1504

Tab. 4: Výsledky algoritmů v druhém prostředí

Na obrázku 26 je zobrazeno druhé testovací prostředí. Výsledky testování algoritmů v druhém prostředí jsou uvedeny v tabulce 4. Opět proběhlo 20 běhů každého algoritmu. V druhém prostředí genetický algoritmus předčil diferenciální evoluci, protože nacházel průměrně lepší řešení v kratším čase. Nejlepší řešení diferenciální evoluce je však lepší než nejlepší nalezené řešení genetickým algoritmem. Nejlepší řešení nalezená oběma algoritmy jsou na obrázcích 27 a 28.



Obr. 27: Nejlepší trasa nalezená genetickým algoritmem v druhém prostředí



Obr. 28: Nejlepší trasa nalezená diferenciální evolucí v druhém prostředí

6 ZÁVĚR

Práce je soustředěna na evoluční výpočetní techniky. V první části práce je lehce naznačena problematika optimalizace jakožto oboru, pod který spadají EVT. Poté jsou vysvětleny teoretické základy EVT a jejich rozdělení.

Pro podrobnější zpracování byly vybrány genetický algoritmus a diferenciální evoluce. V rešeršní části práce jsou představeny obecné rámce obou algoritmů. Cílem práce bylo vybrané algoritmy implementovat a porovnat. Za tímto účelem byl vybrán problém hledání cesty mobilního robota.

Začátek třetí kapitoly tedy obsahuje popis problému navigace mobilního robota. Tento problém má tři stěžejní části: lokalizace, mapování a plánování trasy. Právě poslední část, tedy plánování trasy, byla vybrána jako problém, který bude řešen pomocí genetického algoritmu a diferenciální evoluce. Dále tato kapitola obsahuje dělení různých druhů plánovaní cesty a také reprezentace vnějšího prostředí.

Konkrétně je v práci řešen problém globálního plánovaní trasy ve statickém prostředí, což je obsahem kapitoly čtvrté. Statické prostředí je reprezentováno důmyslným spojením diskrétní a spojité reprezentace prostředí. Tohoto je dosaženo za pomocí programovacího jazyku Python. Virtuální reprezentace je vytvořena za pomocí modulu *Pygame*. Poté jsou představeny základy obou algoritmů jako je účelová funkce, reprezentace jedince či řešení a detekce kolize s překážkami. V závěru kapitoly jsou uvedena vylepšení obecných rámců algoritmů, které bylo potřeba využít, aby byly algoritmy použitelné na zvolený problém.

Oba algoritmy byly porovnány při hledání cesty ve dvou náhodně vygenerovaných prostředích. Každý algoritmus byl spuštěn 20krát v každém prostředí. Dosažené výsledky obou algoritmů byly zprůměrovány a uvedeny v tabulkách 3 a 4. Z výsledků je zřejmé, že genetický algoritmus dosahuje menší časové náročnosti, tedy je schopen dodat uspokojivé řešení v kratším čase. Pro přesnější závěry o srovnání testovaných algoritmů by ovšem bylo potřeba provést větší počet experimentů pro různá nastavení parametrů těchto algoritmů a různě veliká a složitá prostředí.

SEZNAM POUŽITÉ LITERATURY

[1] ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. 1.vyd. Praha: BEN – technická literatura, 2009. ISBN 978-80-7300-218-3.

[2] KOCHENDERFER, Mykel J. a Tim A. WHEELER. Algorithms for optimization [online]. Cambridge, Massachusetts: The MIT Press, 2019 [cit. 2023-05-08]. ISBN 978-026-2039-420. Dostupné z: https://algorithmsbook.com/optimization/files/optimization.pdf

[3] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. Umělá inteligence. Praha: Academia, 2001. ISBN 80-200-0472-6.

[4] International Journal of Computer Information Systems and Industrial Management Applications [online]. MIR Labs, 2017, , 181-205 s. [cit. 2023-05-12]. ISSN 2150-7998.

[5] SLOWIK, Adam a Halina KWASNICKA. Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications [online]. 2020, 32(16), 12363-12379 [cit. 2023-05-10]. ISSN 0941-0643. Dostupné z: doi:10.1007/s00521-020-04832-8

[6] RAZMJOOY, Navid, Mohsen KHALILPOUR a Mehdi RAMEZANI. A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System. Journal of Control, Automation and Electrical Systems [online]. 2016, 27(4), 419-440 [cit. 2023-05-12]. ISSN 2195-3880. Dostupné z: doi:10.1007/s40313-016-0242-6

[7] ESKANDAR, Hadi, Ali SADOLLAH, Ardeshir BAHREININEJAD a Mohd HAMDI. Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems [online]. 2012, 110-111, 151-166 [cit. 2023-05-12]. ISSN 00457949. Dostupné z: doi:10.1016/j.compstruc.2012.07.010

[8] Zombie Survival Optimization: A swarm intelligence algorithm inspired by zombie foraging [online]. Tsukuba, Japan: IEEE, 2013 [cit. 2023-05-12]. ISBN 978-4-9906441-0-9. ISSN 1051-4651. Dostupné z: https://ieeexplore.ieee.org/document/6460301

[9] ZELINKA, Ivan. Umělá inteligence: v problémech globální optimalizace. 1.vyd. Praha: BEN – technická literatura, 2002. ISBN 80-730-0069-5.

[10] KRINK, T., J.S. VESTERSTROM a J. RIGET. Particle swarm optimisation with spatial particle extension. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600) [online]. IEEE, 2002, 1474-1479 [cit. 2023-05-08]. ISBN 0-7803-7282-4. Dostupné z: doi:10.1109/CEC.2002.1004460

[11] XIN YAO, YONG LIU a GUANGMING LIN. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation [online]. 3(2), 82-102 [cit. 2023-05-16]. ISSN 1089778X. Dostupné z: doi:10.1109/4235.771163

[12] GROSS, Tomáš. Plánování cesty mobilního robota. Brno, 2007. Diplomová práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav automatizace a informatiky. Vedoucí práce RNDr. Jiří Dvořák, CSc.

[13] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. Umělá inteligence. Praha: Academia, 2003. ISBN 80-200-1044-0.

[14] NERI, Ferrante a Ville TIRRONEN. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review [online]. 2010, 33(1-2), 61-106 [cit. 2024-04-08]. ISSN 0269-2821. Dostupné z: doi:10.1007/s10462-009-9137-2

[15] KOUBAA, Anis, Hachemi BENNACEUR, Imen CHAARI et al. Introduction to Mobile Robot Path Planning. Robot Path Planning and Cooperation [online]. Cham: Springer International Publishing, 2018, 3-12 [cit. 2024-04-10]. Studies in Computational Intelligence. ISBN 978-3-319-77040-6. Dostupné z: doi:10.1007/978-3-319-77042-0_1

[16] MAŇÁKOVÁ, L. Pokročilé metody plánování cesty mobilního robotu. Brno, 2020, 58 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce RNDr. Jiří Dvořák, CSc.

[17] NumPy. In: The fundamental package for scientific computing with Python [online]. 2024 [cit. 2024-04-16]. Dostupné z: https://numpy.org

[18] HU, Yanrong a S.X. YANG. A knowledge based genetic algorithm for path planning of a mobile robot. IEEE International Conference on Robotics and Automation. Proceedings. ICRA '04. [online]. IEEE, 2022, 4350-43555 [cit. 2023-05-23]. ISBN 0-7803-8232-3. Dostupné z: doi:10.1109/ROBOT.2004.1302402

[19] Intersection of two line segments judged by orientation basic case examples. In: Geeks for geeks [online]. [cit. 2024-04-25]. Dostupné z: https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/

[20] CORMEN, Thomas H. Introduction to algorithms. 3rd ed. Cambridge: MIT Press, 2009, xix, 1292 s. ISBN 978-0-262-03384-8.

[21] How to check if a given point lies inside or outside a polygon? In: Geeks for geeks [online]. 2024 [cit. 2024-05-01]. Dostupné z: https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon

SEZNAM OBRÁZKŮ

SEZNAM TABULEK