Fakulta zemědělská
a technologická
Faculty of Agriculture
and Technology

Jihočeská univerzita
v Českých Budějovicích
University of South Bohemia
in České Budějovice

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
FAKULTA ZEMĚDĚLSKÁ A TECHNOLOGICKÁ

Katedra techniky a kybernetiky

Dizertační práce

Využití genetických algoritmů k optimalizaci činnosti zemědělských strojů v systémech precizního zemědělství

Autor práce: Ing. Mgr. Roman Bumbálek

Vedoucí práce: doc. RNDr. Petr Bartoš, Ph.D.

České Budějovice
2024

Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů

V Českých Budějovicích dne …….. ……..

Podpis

Abstrakt

Metaheuristické metody, inspirované fyzikálními, biologickými, chemickými, sémantickými i sociálními jevy, jsou aplikovány pro optimalizaci úloh s rozsáhlou množinou možných řešení v širokém spektru oborů včetně zemědělství, přičemž významnou část tvoří evoluční výpočty, z nichž jsou pro potřeby precizního zemědělství využívány především genetické algoritmy. Tato práce se zabývá jejich implementací v rámci problematiky plánování tras pojezdu zemědělské techniky s cílem snížení délky trajektorie pohybu na souvratích i ve vnitřní části pozemku. Vytvořený algoritmus se skládá z několika částí, kdy nejprve dochází k výpočtu souřadnic dílčích tras a následně aplikací genetických algoritmů k optimalizaci otáčení souprav na souvratích vhodnou kombinací jejich průjezdů s ohledem na vstupní parametry zahrnující poloměr otáčení a záběr nástroje. Kvalitu získaných výsledků ovlivňuje mnoho parametrů použitého genetického algoritmu, jako je velikost populace, počet generací či zvolený typ selekce, kdy se zvyšujícím se množstvím chromozomů v populaci i generací dochází k vygenerování lepších řešení, umožňujících zkrácení vzdálenosti pojezdu při otáčení na souvratích o více než 37 %, což může přímo vést k úsporám pracovního času, pohonných hmot a nižšímu zhutnění půdy.

Klíčová slova: genetické algoritmy; metaheuristické metody; plánování pojezdových tras; optimalizace trajektorie pohybu; otáčení zemědělských strojů na souvratích

Abstract

Metaheuristic methods, inspired by physical, biological, chemical, semantic and social phenomena, are used to optimise problems with a large number of possible solutions in a wide range of fields, including agriculture, with a significant part of evolutionary computation, of which genetic algorithms are mainly used for precision agriculture. This paper deals with their implementation in the context of the problem of planning the routes of agricultural machines with the aim of reducing the length of the trajectory of movement both on the headlands and in the inner part of the plot. The developed algorithm consists of several parts, where first the coordinates of the partial routes are calculated and then genetic algorithms are applied to optimise the rotation of the implements on the headlands through a suitable combination of their passes with respect to the input parameters, including the turning radius and the tool sweep. The quality of the results obtained is influenced by many factors, such as the size of the population, the number of generations or the type of selection chosen, with better solutions being generated as the number of chromosomes in the population and generations increases, allowing a reduction of more than 37 % in the distance travelled when turning on headlands, which can directly translate into savings in working time, fuel and reduced soil compaction.

Keywords: Genetic algorithms; metaheuristic methods; route planning; trajectory optimisation; headland turning of agricultural machine

Poděkování

Rád bych poděkoval své drahé polovičce Míše za velkou podporu a trpělivost, vedoucímu mé dizertační práce panu doc. RNDr. Petru Bartošovi, Ph.D. za jeho cenné rady udělené nejen v rámci studia, jichž si velice vážím, a také kolegům za jejich ochotu poskytnout pomoc při studiu i práci.

Obsah

Úvod

Zemědělská půda je nepostradatelným statkem, bez kterého by nebyla možná produkce potravin nejen rostlinného, ale potažmo také živočišného původu. I přesto jsme svědky významného zabírání zemědělské půdy pro nezemědělské účely a jejího dalšího znehodnocování v důsledku použití nevhodných technologických postupů. Jedním z nejvýznamnějších typů půdní degradace je zhutnění půdy (Shah et al., 2017). V současnosti je zhutněním ohroženo až 68 milionů hektarů půdy a z toho připadá 33 milionů ha na Evropu (Prăvălie, 2021), přičemž hlavním faktorem zodpovědným za tento stav je především její vysoké mechanické zatížení způsobené pojezdem zemědělské techniky po pozemku (Mileusnić et al., 2022). Současné situaci nepřispívá ani zvyšující se hmotnost zemědělských strojů, která se za posledních několik desetiletí výrazně zvýšila (Shabeb et al., 2021; Sivarajan et al., 2017).

Zhutnění půdy je doprovázeno řadou nežádoucích jevů, jako je zvyšování objemové hmotnosti půdy, snížení velikosti pórů, propustnosti nebo dostupnosti živin, které mají za následek snížení zemědělské produkce (Carla et al, 2022; Cárceles Rodríguez et al., 2022). V důsledku zhutnění klesá schopnost půdy zadržet vodu, čímž se zvyšuje povrchový odtok a riziko vodní eroze (de Lima et al, 2017; Centeri, 2022). Se zhutněním půdy jde ruku v ruce také zvýšení energetické náročnosti obdělávání půdy a snížení jeho kvality (Kumhála et al., 2013).

Mezi významné technologické postupy vedoucí k eliminaci zhutnění půdy patří řízený pohyb zemědělských strojů po pozemcích, tzv. Controlled Traffic Farming (CTF) (Bulgakov et al., 2022). Principiálně je možno CTF charakterizovat tak, že pohyb zemědělských strojů je sveden do předem stanovených drah (Hussein, 2022; Tamirat et al., 2022), které optimálním způsobem pokrývají celý pozemek. Vhodnou volbou trajektorie zemědělských strojů je možno výrazně redukovat míru zhutnění pozemku v důsledku pojezdu zemědělské techniky. Například Edwards et al. (2017) prokázali, že optimalizované dráhy mohou snížit délku tras zemědělských strojů o 18 %, přičemž při využití strojů se stejným rozvorem kol je možno dosáhnout hodnoty až 30 % (Kumhála et al., 2013). Marinello F. et al. (2017) ve své práci konstatují, že díky řízenému pohybu zemědělské mechanizace může být očekáváno zvýšení výnosů plodin o více než 10 %.

Pohyb zemědělské techniky po pozemku lze rozdělit do dvou částí na pojezdy organizovaných dílčích tras ve vnitřní oblasti pole a otáčení na souvrati nutné pro

změnu směru a jejich průjezd, přičemž pro snížení vzdálenosti celkové trajektorie je třeba se zaměřit na optimalizaci pohybu v obou oblastech (Utamima et Djunaidy, 2021). He et al. (2023) uvádí, že snížení pojezdové vzdálenosti při otočkách na souvratích je pro minimalizaci délky kompletní dráhy pohybu velice důležité. Techniky plánování optimálních pojezdových tras zahrnují rozličné přístupy včetně metaheuristických metod zaměřených na hledání optimálních výsledků u problematik s rozsáhlým prostorem možných řešení jako optimalizace rojením částic, optimalizace pomocí mravenčí kolonie či genetické algoritmy (Santos et al., 2020).

V softwarovém inženýrství se pojmem genetický algoritmus označuje takový postup, kdy se opakovaně vybírá, kříží a mutuje populace (data) za účelem vytvoření nové generace, která bude představovat vhodnější řešení daného problému než původní (rodičovská) generace (Han et al., 2017). Součástí rozhodovacího procesu je také stanovení podmínky, při které je možno výpočet ukončit s tím, že bylo nalezeno „dostatečně vhodné“ řešení dané úlohy.

Dizertační práce je zaměřena na aplikaci genetických algoritmů v problematice pojezdu zemědělské techniky po pozemku za účelem snížení délky trajektorií pohybu při otáčení na souvratích. Minimalizace délky dráhy otoček může mít pozitivní vliv na finanční náročnost i ekologickou stopu, kdy snížením pracovního času stroje dochází k úsporám provozních výdajů zahrnujících mzdu obsluhy i množství spotřebovaných pohonných hmot, s čímž se rovněž pojí redukce emisí. Zároveň také dochází k utužení menší části pozemku vedoucí k poklesu povrchového odtoku a tím i zmírnění rizika vodní eroze.

Literární rešerše této práce pojednává o přehledu metaheuristických algoritmů se zaměřením na metody využité v rámci precizního zemědělství, zejména na genetické algoritmy (GA), které jsou blíže popsány. V hlavní části je představen vytvořený algoritmus pro optimalizaci trajektorií pohybu zahrnující dílčí funkce zabývající se výpočtem nejkratších pojezdových tras ve vnitřní části pozemku a implementací jednotlivých operátorů GA jako je inicializační funkce, funkce fitness sloužící k ohodnocení jednotlivých řešení nebo vybrané typy selekce, křížení a mutace. Součástí podrobného popisu všech uvedených algoritmů je také zdrojový kód včetně komentářů vysvětlujících zvolené části.

1 Literární rešerše

1.1 Metaheuristické metody v zemědělství

Metaheuristika je kombinace heuristických metod, jejímž cílem je efektivně podpořit průzkum prohledávaného prostoru, čímž je možné předejít uvíznutí v lokálním minimu ve složitém prostoru. Metaheuristika je inspirována několika tématy, z nichž lze vyzdvihnout například analogii s fyzikálními, chemickými, biologickými, sémantickými a sociálními jevy (Kaur et al. 2020). Klasifikace heuristických metod se u mnohých autorů liší (Ikeda et Inoue, 2016; Kurnia et al., 2018). Darwish et al. (2020), Dhiman et Kumar (2017) a Khishe et Mosavi (2020) používají čtyři základní skupiny: evoluční výpočetní techniku, inteligenci roje, techniku inspirovanou člověkem a společností a fyzikální techniku. Aladeemy et al. (2020) rozdělili metaheuristické metody velmi podobně, ale sloučili skupiny evolučních technik, rojové inteligence a fyzikálně založené metody do jedné velké skupiny nazvané algoritmy inspirované přírodou. Naproti tomu Elshaer et Awad (2020) a De Leon-Aldaco et al. (2015) rozdělili metaheuristické metody pouze do dvou skupin, které nazvali jednoduché a populačně založené. Francik et al. (2020) ve svém přehledu současných výzkumných trendů v oblasti systémů obnovitelných zdrojů energie rovněž zařadili skupiny nazvané evoluční výpočty a rojová inteligence. Hegazy et al. (2018) rozdělili metaheuristické metody do tří skupin na evoluční výpočty, rojová inteligence a metody se základem ve fyzikálních jevech. Na základě podrobné analýzy dostupné literatury zabývající se metaheuristickými metodami a jejich klasifikací je na obrázku 1.1 uveden ucelený přehled důležitých metaheuristických metod. Nejdůležitější metaheuristické metody používané v precizním zemědělství jsou barevně zvýrazněny a podrobně popsány níže. Uvedené metaheuristické metody byly rozděleny do čtyř skupin: evoluční výpočty, rojová inteligence, algoritmy založené na fyzikálních jevech a algoritmy inspirované člověkem.

Do skupiny evolučních výpočtů patří genetické algoritmy (GA), diferenciální evoluce (DE), evoluční strategie (ES), evoluční programování (EP), biogeography-based optimalizátor (BBO), koevoluční algoritmy (Coea), multiobjektivní evoluční algoritmy (Moeas), algoritmus paddy field (PFA), algoritmy odhadu distribuce (Edas), algoritmus opylování květin (FPA), algoritmus klonálního výběru (CSA) a memetický algoritmus (MA).

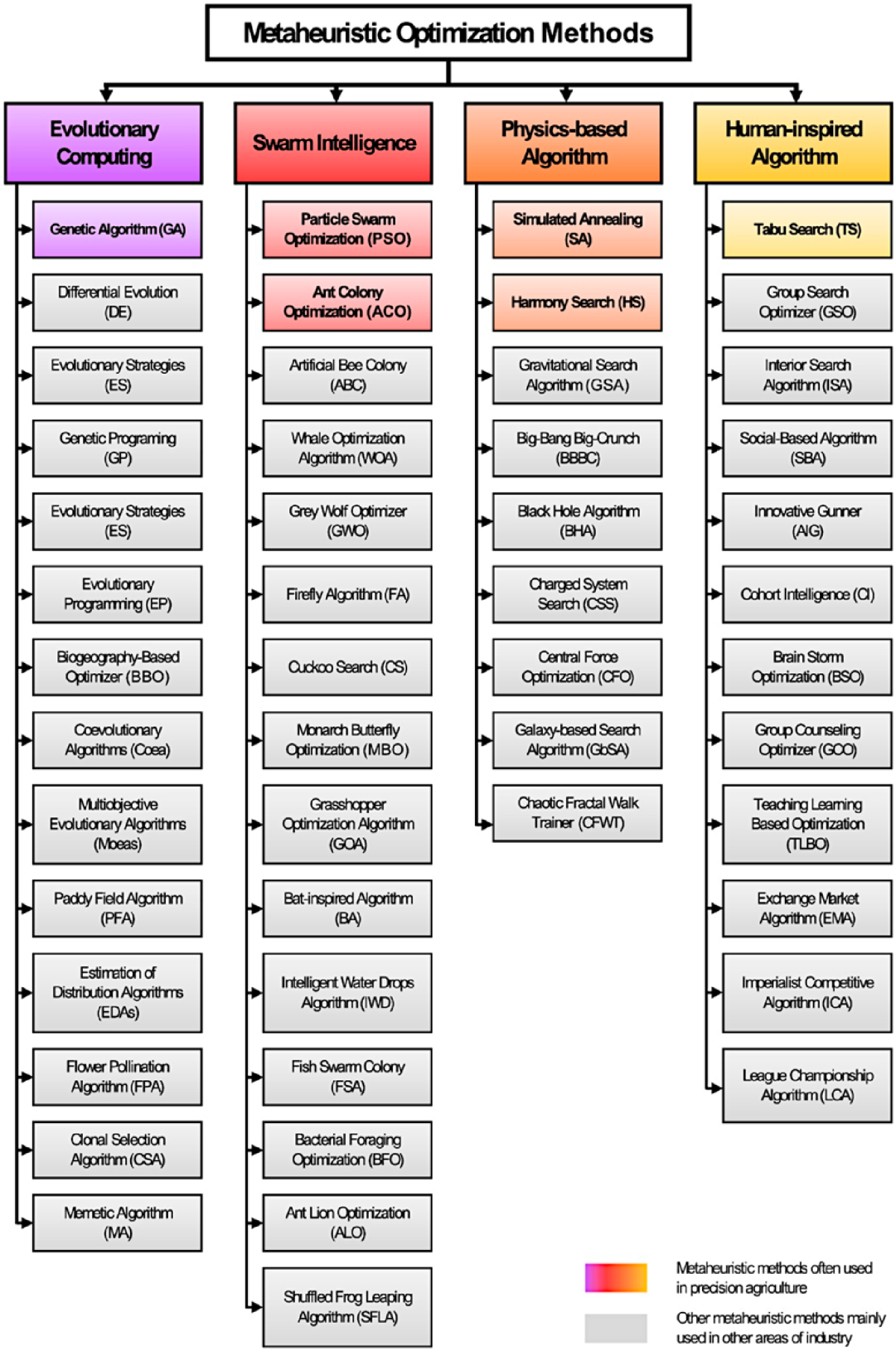
Do skupiny rojové inteligence patří optimalizace rojením částic (PSO), optimalizace pomocí mravenčí kolonie (ACO), algoritmus umělého včelího roje (ABC), velrybí optimalizační algoritmus (WOA), vlčí algoritmus (GWO), algoritmus Firefly (FA), kukaččí algoritmus (CS), optimalizace motýla monarchy (MBO), optimalizační algoritmus lučního koníka (GOA), netopýří algoritmus (BA), algoritmus intelligent water drops (IWD), algoritmus rybího hejna (FSA), optimalizace bacterial foraging (BFO), optimalizace mravkoleva (ALO) a algoritmus žabího skoku (SFLA).

Do skupiny fyzikálně založených algoritmů patří simulované žíhání (SA), harmony search (HS), gravitational search algoritmus (GSA), big-bang big-crunch (BBBC), algoritmus černé díry (BHA), charged system search (CSS), central force optimization (CFO), algoritmus galaxy-based search (GbSA) a chaotic fractal walk trainer (CFWT). Do skupiny algoritmů inspirovaných člověkem patří tabu search (TS), optimalizátor group search (GSO), algoritmus interior search (ISA), social-based algoritmus (SBA), algoritmus innovative gunner (AIG), cohort intelligence (CI), optimalizace brain storm (BSO), optimalizátor group counseling (GCO), optimalizace založená na učení (TLBO), algoritmus burzy (EMA), imperialist competitive algoritmus (ICA) a algoritmus league championship (LCA).

Tabulka 1.1: Přehled nejvyužívanějších metaheuristických metod pro optimalizaci pojezdových tras a problematiku logistiky v zemědělství

Zkratka metody	Název metody
GA	Genetické algoritmy
ACO	Algoritmus mravenčí kolonie
SA	Simulované žíhání
HS	Harmony Search
PSO	Algoritmus optimalizace pomocí roje částic
TS	Tabu Search

Z tabulky 1.1 je patrné, že pro řešení problémů plánování tras pojezdu a logistiky zemědělských polí bylo v literatuře využito několik metaheuristických metod (Bochtis et Vougioukas, 2008). Nejčastější algoritmy aplikované na tento problém jsou genetické algoritmy, algoritmus mravenčí kolonie, simulované žíhání, harmony search, optimalizace pomocí roje částic a tabu search.



Obrázek 1.1: Klasifikace metaheuristických optimalizačních metod

1.1.1 Algoritmus mravenčí kolonie (ACO)

Algoritmus mravenčí kolonie je optimalizační metoda, která vychází z chování mravenců hledajících potravu v okolí svého hnízda. Pohyb mravenců je náhodný, avšak pro návrat zpět do výchozí pozice využívají stejnou cestu, na které v případě nalezení potravy zanechávají feromony, a tak se cesta stává atraktivnější pro ostatní mravence, jenž vypouštěné feromony následují (Joo et Lim, 2018). Tento typ komunikace je nazýván nepřímým a je řízen intenzitou feromonové dráhy ovlivněnou počtem a frekvencí průchodů mravenců, trasa využívaná méně v průběhu času přestává být pro mravence zajímavá z důvodu vypařování feromonů (Pu et al., 2020). Základní princip této metody spočívá v nulové počáteční koncentraci feromonů na všech cestách, tedy pravděpodobnost výběru je pro všechny trasy stejná, vzhledem ke stejné rychlosti pohybu mravenců se ti, kteří si zvolí kratší cestu, dostanou k potravě rychleji, a proto začnou uvolňovat feromony na zpáteční cestě dříve. Kratší cesta je tedy dříve pokryta feromony a začne přitahovat více mravenců, pravděpodobnost výběru je úměrná intenzitě feromonů na cestě v aktuálním čase. Se zvyšující se intenzitou se během několika iterací zvyšuje i pravděpodobnost výběru nejkratší cesty (Feng, 2020). Mezi hlavní parametry algoritmu patří koncentrace feromonů, maximální doba iterace, velikost populace, počet proměnných a stupeň důležitosti feromonů (Pamosoaji et Setvohadi, 2020).

Přístup založený na optimalizaci mravenčí kolonie pro generování optimálního pokryvu pole pojezdovými trasami zemědělské techniky pro sklizňové operace byl použit v práci Bakhtiari et al. (2013). Podstatou jejich řešení je pojezd v podobě B-vzoru, což je výsledek kombinatorického optimalizačního procesu minimalizující celkovou délku pojezdové trasy či operační čas, kdy sklízecí mlátička vykládá zrno do stacionárního zařízení umístěného mimo oblast pole nebo na jeho hranici. Výsledky vycházející z porovnání optimálních plánů generovaných vybranými operátory s konvenčními plány ukazují snížení nepracovní vzdálenosti na poli (celková délka manévrů prováděných sklízecí mlátičkou při obratech na souvratích) v rozmezí 19,3-42,1 %, zatímco úspora celkové nepracovní vzdálenosti byla v rozmezí 18-43,8 %.

Feng (2020) se zabýval logistickým modelem potenciálně použitelným v precizním zemědělství na základě algoritmu mravenčí kolonie, k jehož hlavním výhodám řadil univerzálnost a absenci potřeby počáteční trasy. Dále také uvádí, že ACO není závislý na volbě dílčí počáteční trasy a také ji není třeba v procesu vyhledávání ručně

upravovat. Algoritmus mravenčí kolonie lze využít také v případě elektrických vozidel, jejichž význam v zemědělství v budoucnu výrazně vzroste (Mavrovouniotis et al., 2018), například Joo et Lim (2018) představili efektivní metodu energeticky účinného směrování s využitím ACO pro maximalizaci energetické účinnosti (Alaiso et al., 2013). Optimalizací efektivity přepravy produktů na konkrétní místa a zlepšením manipulační kapacity se zabývali Mutar et al. (2020), jejichž výzkum prokázal značný potenciál navržené ACO metody, vzhledem k získání lepších výsledků než u jiných algoritmů (např. SA).

1.1.2 Simulované žíhání (SA)

Simulované žíhání je stochastický algoritmus vytvořený Kirkpatrickem, Gelettem a Vechchim v roce 1983 (Avdemir et Karagul, 2020), který je aplikován k nalezení optimálního řešení při hledání globálních extrémů funkce s lokálními minimy (Mohiuddin et al., 2014). SA je inspirováno fyzikálním procesem žíhání kovů (Grabusts et al., 2019), kdy s vysokou počáteční teplotou materiálu jeho částice získají dostatečné množství energie k úniku z výchozích poloh (lokálních minim) a postupným ochlazováním dochází k nastolení tepelné rovnováhy. K základním parametrům SA se řadí počáteční teplota, rychlost ochlazování a podmínky ukončení procesu (Vahdanjoo et al., 2020). Vnitřní část algoritmu tvoří Metropolisův algoritmus, který generuje nová řešení drobnou úpravou či posunem pravděpodobnostní funkce stávajícího řešení, nejlepší dosažené řešení je určeno během iteračních kroků (Mohiuddin et al., 2014; Grabusts et al., 2019; Vahdanjoo et al., 2020).

Cerdeira-Pena et al. (2017) implementují a testují dva odlišné heuristické algoritmy založené na metodě Tabu Search a principu Simulovaného Žíhání, vytvořený model aplikují na reálný problém optimalizace pojezdových tras několika sklízecích mlátiček. Získané výsledky ukázaly, že takto upravená SA heuristika může být úspěšně využita jako součást nástroje pro řízení zemědělských operací. Grabusts et al. (2019) použili heuristiku SA pro detekci optimální trasy mezi objekty s využitím polohy GPS a vyvinuli software pro hledání a optimalizaci nejkratší dráhy pojezdu mezi různými objekty. Moriguchi (2020) algoritmus simulovaného žíhání vylepšil z hlediska rychlosti a spolehlivosti výpočtů a aplikoval ho při optimalizaci prořezávky lesního porostu. V precizním zemědělství se také stávají populární malá bezpilotní letadla a drony, jež vyžadují sofistikované plánování letové dráhy, které lze též pomocí SA optimalizovat (Behnck et al., 2015). Simulované žíhání bylo také úspěšně použito pro monitorování zemědělství. Například Leitold et al. (2018) navrhli metodiku založenou

na Simulovaném žíhání pro přiřazení dalších senzorů k dynamickým monitorovacím systémům. Bochtis et Vougioukas (2008) navrhli přístup založený na SA, který využili k řešení problému plánování trasy pro aplikaci herbicidů, celková ujetá vzdálenost po vybraném pozemku byla 335,767 m a vzdálenost ujetá na souvratích 95,767 m. Metoda navržená Conesou-Muñozem et al. (2016) ukázala celkovou ujetou vzdálenost na stejném pozemku 334,439 m a vzdálenost ujetou na souvratích 94,439 m. Porovnáním obou výsledků lze konstatovat, že rozdíly mezi vypočtenými vzdálenostmi jsou menší než 1,4 %.

1.1.3 Harmony search

Tato optimalizační metoda je stejně jako mnoho jiných metaheuristických metod inspirována procesy odehrávajícími se v reálném světě, konkrétně procesem hudební improvizace při skládání melodie (Liu et al., 2020). Počáteční harmonie je generována jako soubor náhodných not, které jsou uložené do paměti. Vytvořené noty jsou laděny do nové harmonie malým posunem jejich tónů (Valente et al., 2013). V každé iteraci algoritmu je vytvořena nová harmonie, mutace jednotlivých proměnných (not) jsou realizovány jejich drobnou změnou či jejich úplným nahrazením, na konci každé iterace probíhá výběr harmonie a aktualizace paměti. Možnost zohlednění každého člena harmonie při generování nových řešení, zachování přesnosti, pokud jsou proměnné součástí improvizačního procesu, a náhodná inicializace rozhodovacích proměnných náleží k hlavním výhodám metody Hormony Search (Mahaleh et Mirroshandel, 2018).

Pravděpodobně nejčastější využití HS v zemědělství nachází při plánování pojezdových tras, například Utamina et al. (2019) použili tento algoritmus pro řešení problému optimální trajektorie pohybu zemědělské techniky v rámci logistiky na poli. Stejně jako jiné heuristiky byl i algoritmus Harmony Search aplikován pro optimalizaci pohybu bezpilotních letounů v rámci řízení precizního zemědělství a plánování tras v robotice (Valente et al., 2013; Mahaleh et Mirroshandel, 2018). Výzkumníci vynakládají velké úsilí na zdokonalení stávajících algoritmů HS. Například Liu et al. (2020) představili metodu využívající algoritmus HS založený na kódování pomocí přirozených čísel pro problematiku plánování tras, jenž při zpracování přirozených čísel dosáhla vysoké účinnosti a poskytla lepší výsledky ve srovnání s jinými testovanými typy algoritmu HS. Valente et al. (2013) vyvinuli postup využívající algoritmus Harmony Search k optimalizaci letových drah, jenž vykázal lepší výsledky při optimalizaci trasy ve srovnání s metodou prezentovanou Barrientosem et al. (2011) díky snížení počtu otáčení v trajektoriích pokrývajících pozemek a dodržení stanovené výchozí

a cílové polohy. Výpočetní čas je ve srovnání s předchozím přístupem delší, to však lze považovat za přijatelnou nevýhodu vzhledem k tomu, že cílem vytvořeného plánovače tras není práce v reálném čase.

1.1.4 Algoritmus optimalizace pomocí roje částic (PSO)

Algoritmus optimalizace pomocí roje částic, který je založený na prohledávání multidimensionálního prostoru rojem částic určených k nalezení globálně optimálního řešení daného problému, byl poprvé představen Kennedym a Eberhartem (1995). Rychlost částic se přepočítává na základě znalosti jejich dosavadních fitness hodnot nebo zkušeností ostatních členů roje (Das et Jena, 2020). Algoritmus PSO je široce používán v praktických aplikacích a teoretickém výzkumu plánování cesty mobilních agentů díky svým silným vyhledávacím schopnostem, rychlé konvergenci a vysoké efektivitě. Výhodou optimalizace roje částic je vysoká rychlost prohledávání, nízký počet parametrů, jednoduchá struktura a snadnější implementace ve fázi ověřování. K nevýhodám se řadí nízká přesnost konvergence, eventualita snadného uvíznutí v lokálním minimu a špatná robustnost (Li et al., 2020).

Metodu optimalizace pomocí roje částic lze využít v zemědělství 4.0 například pro plánování kapacity servisních služeb pro zemědělské stroje (Hu et al., 2020) a také pro trasy bezpilotních letounů. Mukherjee et al. (2020) použili rojovou optimalizaci v úlohách náročných na zpracování, jako je vizuální identifikace zemědělských pozemků a sledování zdravotního stavu či růstu plodin. Pravděpodobně nejdůležitější oblastí s přesahem do precizního zemědělství, kde se PSO využívá, je robotika. Das et Jena (2020) a Li et al. (2020) použili algoritmus PSO pro plánování dráhy mobilních robotů.

1.1.5 Tabu Search

Algoritmus Tabu Search uvedený v roce 1989 F. Gloverem, je určený pro řešení kombinatorických problémů vyskytujících se v reálných situacích oblasti plánování (Glover, 1989) a optimalizaci logistických problémů (Seyyedhasani et Dvorak, 2017). Metoda Tabu Search je založena na lokálním prohledáváním množiny řešení, při kterém vzniká Tabu List obsahující seznam již zpracovaných řešení, k nimž by se algoritmus již neměl vracet, který zabraňuje uvíznutí v lokálním minimu (Seyyedhasani et Dvorak, 2017; Soto et al., 2017). Tabu List má pevnou délku n, jež je vstupním parametrem algoritmu, a obsahuje seznam posledních n změn v podobě dvojic prvků [proměnná;

hodnota], při výběru hodnoty proměnné je zjišťováno, zda tato dvojice již není zaznamenána v Tabu. Stav zakázaných, již provedených, změn v závislosti na čase a okolnostech je založen na vyvíjející se paměti. V algoritmech implementujících Tabu List však existuje funkce, která umožňuje výběr řešení, i když je řešení v Tabu seznamu, nazývající se aspirační kritérium (He et al., 2020; Xing et al., 2020). Výběr nežádoucí změny může nastat v případě dosavadního nenalezení lepšího řešení (Moon et al., 2016). Tabu Search dosahuje lepších výsledků při využití několika lokálních vyhledávacích prostorů, ale ověřování hodnot průběžných řešení v Tabu seznamu je časově náročnější (He et al., 2019; Seyyedhasani et Dvorak, 2019).

Stejně jako jiné algoritmy je pravděpodobně nejčastější použití algoritmu Tabu Search v zemědělství spojeno s problémem plánování zemědělských tras. Například Utamima et al. (2019) použili algoritmus TS společně s algoritmem HS pro plánování pojezdu techniky v polní logistice. Sethanan et al. (2013) navrhli matematický model pro optimalizaci mechanizované sklizně cukrové třtiny s cílem maximalizovat procento výnosu cukrové třtiny. K řešení modelu byly použity dva heuristické algoritmy. První algoritmus byl zaměřen na plánování mechanizované sklizně, zatímco druhý algoritmus byl zaměřen na optimalizaci řešení prvního algoritmu pomocí Tabu Search. Dle výsledků bylo prokázáno 16,38% zlepšení v produkci cukru. He et al. (2018) poskytli operační model pro stanovení optimálního rozvržení sklízecích mlátiček pro nespojité zemědělské pozemky. Ke stanovení optimálního plánu pro dosažení minimalizace celkové doby sklizně i doby sklizně na jednotlivých pozemcích použili hybridní metodu Tabu Search. Hybridní algoritmus využívající jak adaptivní prohledávání velkého okolí, tak Tabu Search byl také použit ke snížení nepracovní vzdálenosti v polní logistice pro heterogenní sklízecí stroje. Seyyedhasani et al. (2019) představili výběr směrovacího algoritmu pro plánování pokrytí pole pomocí dvou směrovacích algoritmů, Clarke-Wrightova algoritmu a algoritmu Tabu Search a v rámci svého výzkumu došli k závěru, že algoritmus TS poskytuje lepší výsledek ve srovnání s algoritmem Clarke-Wright. Kong et al. (2019) vyvinuli na základě algoritmu Tabu Search citlivý optimalizačně aplikační rámec pro rozhodování v precizním zemědělství a matematický model ke zpracování dat v reálném čase, poté provedli aplikaci na hypotetickou případovou studii k optimalizaci sklizně cukrové třtiny. Edwards et al. (2015) navrhli nový plánovací algoritmus pracující s individuálními pracovními plány pro více strojů provádějících více po sobě jdoucích operací na více polích. Využili dva optimalizační algoritmy včetně standardního Tabu Search.

1.2 Genetické algoritmy

Genetické algoritmy se řadí mezi heuristické metody. Jejich základní myšlenka je inspirována biologickou evolucí, fungují tedy na principu mechanismu přirozeného výběru a genetiky (Ferreira Neto et al., 2011, Ikeda et Inoue, 2016, Kurnia et al., 2018). Vývoj genetických algoritmů započal již v šedesátých letech 20. století a je spojen s osobností J. Hollanda (Lamini et al., 2018, Elhoseny et al., 2018). Za milník této problematiky je považován rok 1975, kdy J. Holland předložil svůj výzkum v publikaci *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (Qiongbing et Lixin, 2016, Sales et al., 2018). V současné době jsou genetické algoritmy využívané pro optimalizaci modelů a hledání optimálních řešení rozsáhlých a složitých problémů (Alipour et al., 2018, Mohammed et al., 2017) v mnoha oblastech, jako strojírenství, informační technologie, ekonomie a řízení dodavatelských řetězců (Dao et al., 2017, Sales et al., 2018). Také v zemědělství je možné tyto algoritmy uplatnit, například pro optimalizaci pojezdu zemědělské techniky po pozemku (Neungmatcha et al., 2013, Tong et al., 2017, Gracia et al., 2013), rozčlenění pozemku (Ferreira Neto et al., 2011, Sales et al., 2018) a modelování predikce a optimalizace výnosů (Ali et al., 2018, Hilal et al., 2018). Mohammed et al. (2017) uvádí, že genetické algoritmy lze efektivně využít na hledání optimálního řešení, pokud je prostor všech řešení příliš velký a lineární programování nedokáže nalézt teoretické řešení ve vyhovujícím čase a na řešení problémů s více omezeními. Mezi hlavní výhody patří možnost optimalizovat problémy s mnoho řešeními, využívání paralelního výběru dat, což eliminuje uvíznutí v lokálně optimálním řešení, a dobrá srozumitelnost algoritmu (Ngoc et al., 2014).

1.2.1 Struktura GA

Lamini et al. (2018) uvádí, že genetický algoritmus obsahuje pět základních částí:

- Výběr hodnot proměnných parametrů genetického algoritmu, jako velikost populace, pravděpodobnost křížení, pravděpodobnost mutace a kritérium k zastavení algoritmu
- Vhodné kódovaní dat
- Metodu generování počáteční populace
- Funkci fitness sloužící k vyhodnocení kvality každého potenciálního řešení
- Genetické operátory, které upravují genetické složení rodičovských chromozomů k vytvoření nových chromozomů (potomstva)

Ukázka pseudokódu klasického genetického algoritmu (Lamini et al., 2018):

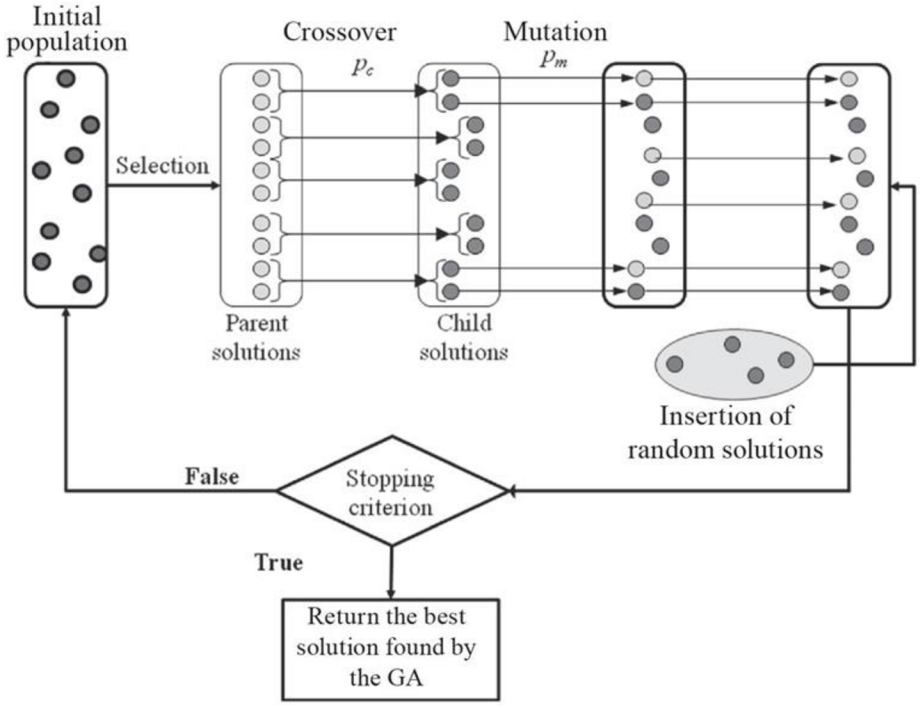
1: **Vstup:** N : Velikost populace; P_c : Pravděpodobnost křížení; P_m : Pravděpodobnost mutace
2: **Výstup:** Nejlepší chromozom
3: $t \leftarrow 0$
4: Náhodné vytvoření počáteční populace $P(t)$
5: **zatímco** (není to podmínka k ukončení) **proveď**
6: Vyhodnocení $P(t)$ použitím fitness funkce
7: Selekce $P(t)$ z $P(t-1)$
8: Rekombinace $P(t)$
9: Vyhodnocení $P(t)$
10: Nahrazení $P(t)$ za $P(t-1)$
11: $t \leftarrow t+1$
12: **konec**

V prvním kroku procesu genetického algoritmu probíhá kódování dat do chromozomů a vygenerování náhodné populace. Chromozomy mohou být tvořeny binárním kódem (Elhoseny et al., 2018, Yi et al., 2016), celými i reálnými čísly (Tong et al., 2017, Han et al., 2017, Alipour et al., 2018, Ikeda et Inoue, 2016, Amal et al., 2018, Kurnia et al., 2018, Sun et al., 2018), což je dle Elferchichi et al. (2009) pro reálné hodnoty dat vhodnější způsob než binární kódování, protože proces kódování potřebuje velké množství počítačové paměti a mnoho výpočetního času. Data v jednotlivých chromozomech mohou být také reprezentovány souřadnicemi bodů (Nazarahari et al., 2019, Lee et al., 2018, Kwaśniewski et Gosiewski, 2018).

Poté je každý chromozom ohodnocen pomocí funkce fitness, která na základě vybraného kritéria nebo více kritérií vypočítá kvalitu chromozomu, označovanou také jako hodnotu fitness, jenž je důležitým kritériem pro další krok, což je výběr (selekce). Mařík et al. (2001) uvádí, že úkolem selekce je upřednostňovat kvalitnější jedince před horšími. Z vybraných chromozomů (rodiče) vznikne využitím dalších genetických operátorů, což je křížení a mutace, nová generace chromozomů (potomci), pro něž je vypočítána kvalita, následně jejich zařazením vzniká nová generace populace. Tento proces se opakuje, dokud nenastane zlepšení kvality, které by splňovalo optimalizační kritérium (Koca et al., 2018). Dle Ferreira Neto et al. (2011) typický cyklus genetického algoritmu obsahuje 50 až 500 generací, avšak může být i vyšší. Populace na konci tohoto cyklu obsahuje jeden nebo více chromozomů s vysokou kvalitou.

Dao et al. (2017) uvádí, že od počátku představení genetických algoritmů před více než čtyřmi dekádami, zaznamenaly jednotlivé části genetických algoritmů stejně

jako jejich struktura značný vývoj, díky čemuž jsou dnes výrazně výkonnější než tradiční genetické algoritmy. Dále také autor říká, že při využití genetických algoritmů je třeba přizpůsobit některé části, například kódování chromozomů, křížení nebo mutaci dané problematice. Na obrázku 1.2 se nachází schéma genetického algoritmu, který je modifikován vložením postupu obnovení populace určeným ke vkládání nových, náhodně vygenerovaných chromozomů do každé nové generace, což má zajistit velkou rozmanitost členů populace a vyloučit předčasnou konvergenci řešení. Výkon genetických algoritmů také velmi závisí na nastavení parametrů, zejména pravděpodobnosti mutace a křížení (Neungmatcha et al., 2013).



Obrázek 1.2: Proces upraveného genetického algoritmu, převzato z Ferreira Neto et al. (2011)

1.2.2 Selekce

Selekce je velmi důležitým genetickým operátorem, používaným k výběru chromozomů, které budou mít v následujícím křížení roli rodiče, jenž poslouží ke vzniku nových chromozomů, potomků, kteří budou součástí další generace (Mařík et al., 2001, Elhoseny et al., 2018). Pravděpodobnost výběru je vyšší u kvalitních, dobře hodnocených chromozomů, což může zlepšit průměrnou kvalitu nové generace populace. Vybraný způsob selekce má tedy přímý vliv na výsledky genetického algoritmu, a proto je podstatné poskytnout vhodný selekční tlak, jelikož v opačném případě může nastat zastavení vývoje populace nebo ztráta její rozmanitosti (Lamini et al., 2018, Han et al.,

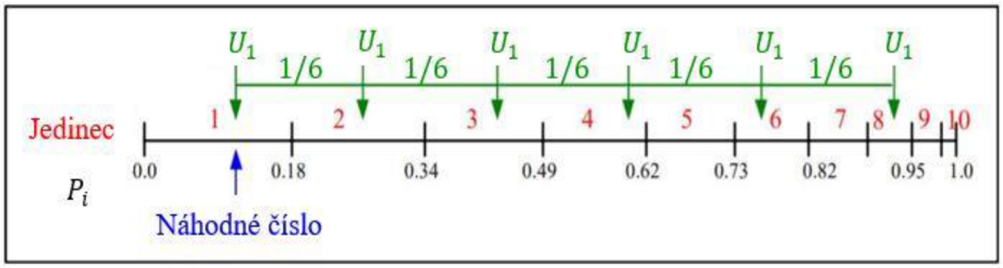
2017, Qiongbing et Lixin, 2016). Selekčních metod vyžívaných v genetických algoritmech je více, mezi nejpoužívanější patří:

- **Pořadová selekce** spočívá v sestupném seřazení všech jedinců dle jejich kvality a přiřazením hodnoty od 1 do N, což je celkový počet jedinců, tak, že nejlepší jedinec získá hodnotu N a nejhorší hodnotu 1. Nejlepší jedinec má tedy N-násobně větší pravděpodobnost výběru než nejhorší jedinec (Kwaśniewski et Gosiewski, 2018, Kurnia et al., 2018).

- **Elitářský výběr** určí jednoho nebo více jedinců s nejvyšší hodnotou, kteří jsou zařazeni do nové generaci, čímž je zajištěno přežití nejkvalitnějších jedinců. Zbylé chromozomy mohou projít další selekční metodou nebo jsou rovnou reprodukováni pomocí křížení a mutace (Lamini et al., 2018, Lee et al., 2018, Kwaśniewski et Gosiewski, 2018, Ikeda et Inoue, 2016, Gao et al., 2017, Amal et al., 2018).

- **Turnajová metoda** nejprve náhodně určí skupiny minimálně dvou jedinců, v těch poté mezi sebou porovnává jednotlivé chromozomy a dle hodnoty fitness určí vítěze turnaje (skupiny), tedy chromozom postupující k další rekombinaci (Elferchichi et al., 2009, Sales et al., 2018, Assaf et Saleh, 2017).

- **Stochastické univerzální vzorkování** je selekční metoda vycházející z ruletové metody s tím rozdílem, že výběr požadovaného počtu jedinců N_p probíhá najednou. Nejprve se náhodně vygeneruje ukazatel U_1, který leží v intervalu $\langle 0\,;\, 1/N_p \rangle$, další ukazatele $U_i, i = 2 \ldots N_p$ jsou od U_1 ve vzdálenosti $1/N_p \cdot (i-1)$. Pokud například $N_p = 6$, pak pro náhodně vygenerované U_1 platí, že $U_1 \in \langle 0; 1/6 \rangle$, pro ostatní ukazatele platí, že $U_i = U_1 + i \cdot 1/6$, sousední ukazatelé budou tedy od sebe vzdáleni o $1/6$, viz obrázek 1.3 (Wieczorek et Ignaciuk, 2018, Yi et al., 2016, Sun et al., 2018, Polheim, 2006).

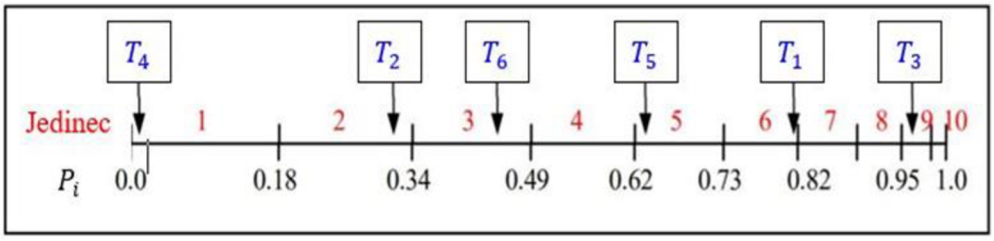


Obrázek 1.3: Znázornění stochastické univerzální metody, upraveno z Polheim (2006)

– **Mechanismus ruletového kola** vybírá jedince s pravděpodobností, která je přímo úměrná jeho kvalitě. Pravděpodobnost výběru jedince P_i je rovna podílu jeho hodnoty fitness a celkovému součtu fitness hodnot celé populace, což lze zapsat pomocí vzorce 1.1, kde f_i je fitness hodnota i-tého jedince a Q je počet členů celé populace:

$$P_i = \frac{f_i}{\sum_{k=1}^{Q} f_k}. \tag{1.1}$$

Výběr požadovaného počtu jedinců N_p probíhá vygenerováním N náhodných čísel T_i z intervalu $\langle 0; 1 \rangle$. Tato metoda svůj název získala připodobněním k principu ruletového kola, na kterém ale nemají všichni jedinci výseč o stejné velikosti, nýbrž každý jedinec má výseč, jehož velikost je úměrná pravděpodobnosti jeho výběru. Generování náhodných čísel T_i znázorňuje N otočení rulety. Interpretace mechanismu ruletového kola je možná i pomocí číselné osy o délce 1, na které budou vyneseny hodnoty pravděpodobností výběru (Nazarahari et al., 2019, Han et al., 2017, Qiongbing et Lixin, 2016, Villar et al., 2016, Alipour et al., 2018, Ngoc et al., 2014, Tong et al., 2017, Ikeda et Inoue, 2016, Gao et al., 2017, Gracia et al., 2013, Mohammed et al., 2017, Polheim, 2006). Na obrázku 1.4 je znázorněna situace, kdy $Q = 10$ a $N_p = 6$.

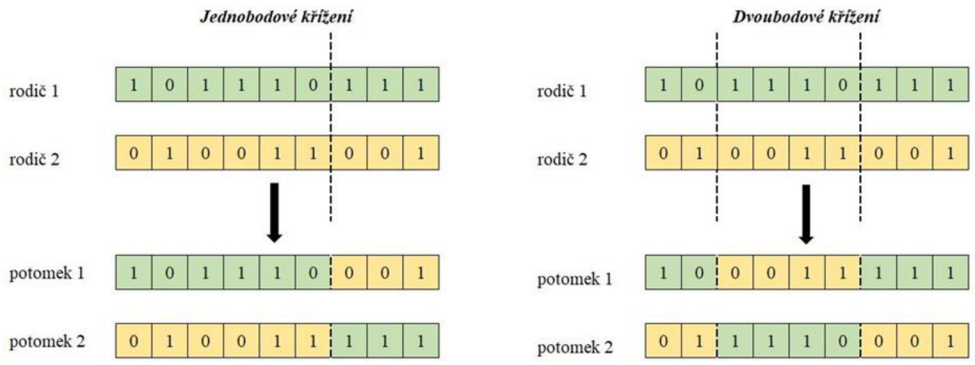


Obrázek 1.4: Znázornění mechanismu ruletového kola, upraveno z Polheim (2006)

1.2.3 Křížení

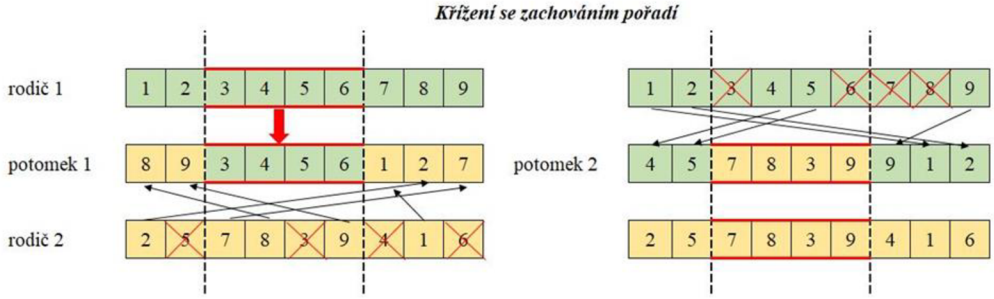
Křížení slouží k vytvoření chromozomů, takzvaných potomků, s novou kombinací genů pomocí smísení genetických informací tradičně dvou vybraných rodičovských chromozomů ze stávající populace. Tento proces udržuje rozmanitost a zvyšuje kvalitu možných řešení. Hlavním cílem křížení je vytvořit kvalitnější jedince a tím vést populaci chromozomů k přiblížení se k optimálnímu výsledku (Elhoseny et al., 2018, Nazahari et al., 2019, Assaf et Saleh, 2017). Metod křížení existuje poměrně značné množství, k hlavním z nich se řadí:

- **Jednobodové křížení (one-point crossover)** je nejjednodušší metodou křížení, kdy ze dvou rodičovských chromozomů vznikají dva potomci. Prvním krokem tohoto procesu je náhodné zvolení křížového bodu, což je celé číslo o hodnotě v rozsahu 1 až $N - 1$, kde N značí počet genů v chromozomu rodiče. Poté se v tomto místě oba rodiče rozdělí na dvě části. První potomek je složen z genů levé části prvního rodiče a pravé části druhého rodiče. Druhý potomek je tvořen opačně (Lee et al., 2018, Notte et al., 2016, Baniamerian et al., 2019, Amal et al., 2018).
- **Dvoubodové a vícebodové křížení (two-point and multi-point crossover)** je obdoba jednobodového křížení, avšak jak už vyplývá z názvu, má dva či více křížových bodů, které jsou opět voleny náhodně. Potomci získávají geny kombinací řetězců, které jsou v rodičovských chromozomech vymezeny pomocí křížových bodů. Princip jednobodového a dvoubodové křížení je znázorněn na obrázku 1.5. Výhodou oproti jednobodovému křížení je vyšší rozmanitost potomků a také možnost zavedení více rodičovských chromozomů (Han et al., 2017, Villar et al., 2016, Wieczorek et Ignaciuk, 2018, Sun et al., 2018).



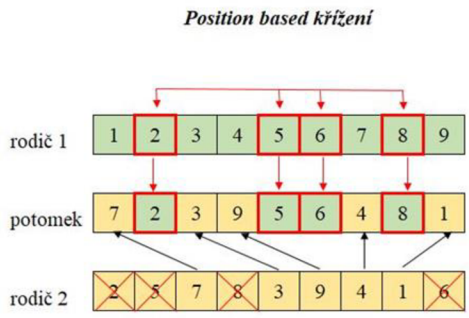
Obrázek 1.5: Znázornění jednobodového a dvoubodového křížení

– **Křížení se zachováním pořadí (Order crossover)** se zakládá na vytvoření potomků se zachováním relativního pořadí genů z rodičů. Rodiče se nejprve pomocí dvou náhodně vybraných křížících bodů rozdělí na tři části. Prostřední řetězec genů jednoho rodiče se bez změny přenese na stejnou pozici do potomka, zbývající geny se poté doplní z druhého rodiče tak, že geny, které jsou již na potomka přeneseny od prvního rodiče, se vynechávají, viz obr 1.6. Při doplnění zbývajících genů se začínají prvky rodiče vybírat od druhého bodu křížení a do potomka se zařazují také od druhého bodu křížení, po doplnění třetí části potomka se doplní část před prvním bodem křížení (Gracia et al., 2013, Mařík et al., 2001, Alipour et al., 2018).



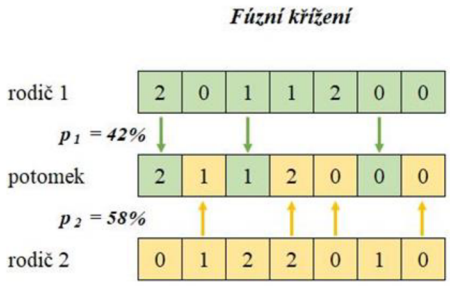
Obrázek 1.6: Znázornění křížení se zachováním pořadí

– **Křížení na základě pozice (Position based crossover)** nejprve náhodně vybere několik genů jednoho z rodičů a překopíruje je z jejich pozic na stejné pozice v řetězci potomka. Vybrané geny jsou z druhého rodiče vymazány a jeho zbývající geny jsou postupně od prvního doplňovány do prázdných pozic potomka. K vytvoření druhého potomka se role rodičů obracejí a celý proces probíhá znovu (Kurnia et al., 2018, Kellegöz et al., 2008). Tento způsob křížení je znázorněn na obrázku 1.7.



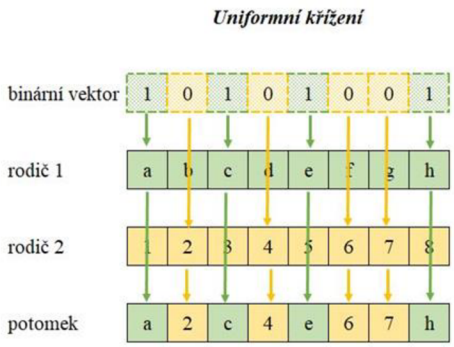
Obrázek 1.7: Znázornění křížení na základě pozice, upraveno z Kellegöz et al., (2008)

– **Fúzní křížení (Fusion crossover)** spočívá ve vytvoření jednoho potomka z genů dvou rodičů, kde kombinace jejich genů je ovlivněna jejich fitness hodnotou. Předpokládá se, že předání určitého genu z kvalitnějšího rodiče více přispěje k celkové kvalitě potomka. Pro každý gen rodiče 1 s fitness hodnotou f_1 je pravděpodobnost předání $p_1 = f_1/(f_1 + f_2)$, pro každý gen rodiče 2 s fitness hodnotou f_2 je pravděpodobnost předání $p_2 = f_2/(f_1 + f_2)$. Geny, které mají oba rodiče stejné, se předávají dále, pokud geny stejné nejsou, je pravděpodobnost jejich předání potomkovi větší u genu kvalitnějšího rodiče (Sales et al., 2018, Beasly et Chu, 1995). Na obrázku 1.8 je znázorněn proces fúzního křížení, kdy $p_1 = 42\%$ a $p_2 = 58\%$.



Obrázek 1.8:Znázornění fúzního křížení, upraveno z Sales et al. (2018)

– **Uniformní křížení (Uniform crossover)** předává potomkovi geny rodičů pomocí náhodně vygenerovaného binárního vektoru. Při počtu N genů v každém rodiči má binární vektor délku N. Pokud hodnota na i-tém místě vektoru, $i \in \langle 1; N \rangle$, je rovna 1, je na i-té místo potomka předán gen prvního rodiče, pokud je rovna 0, předává se se gen druhého rodiče, viz obrázek 1.9 (Mohammed et al., 2017, Yi et al., 2016). Ikeda et Inoue (2016) ve své práci tento způsob křížení označují jako hybridní.

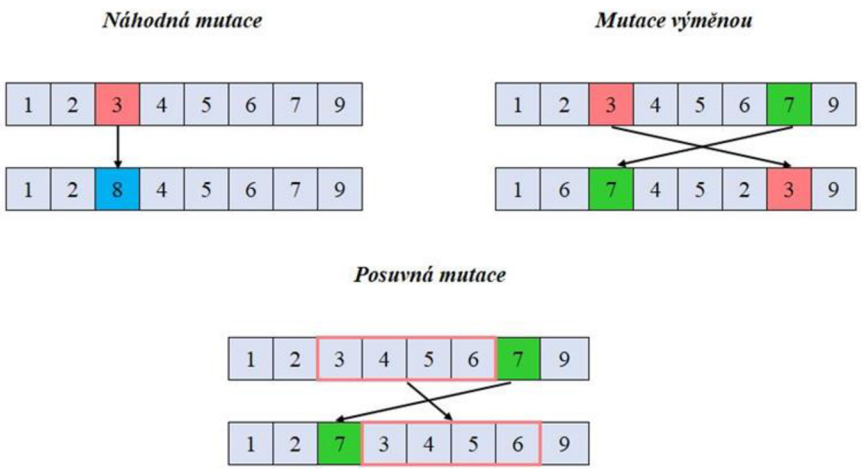


Obrázek 1.9: Znázornění uniformního křížení, upraveno z Mohammed et al. (2017)

K dalším metodám křížení se řadí křížení s částečným zobrazením, cyklické křížení, křížení s rekombinací hran (Lamini et al., 2018, Alipour et al., 2018), smart multipoint crossover (Alipour et al., 2018), same adjency crossover (Qiongbing et Lixin, 2016) a aritmetické křížení (Elferchichi et al., 2009, Kwaśniewski et Gosiewski, 2018, Nazarahari et al., 2019).

1.2.4 Mutace

Tento genetický operátor následuje za křížením a provádí náhodné změny jednoho či více genů nebo úpravy jejich pořadí v chromozomech. Výběr genu určeného k mutaci se řídí pravděpodobností mutace P_m (Neungmatcha et al., 2013, Kurnia et al., 2018, Wieczorek et Ignaciuk, 2018). Pomocí mutace je možné zajistit v generacích rozmanitost řetězců chromozomů, snížit míru konvergence a předejít problému s řešením vyskytujícím se v oblasti lokálního optima (Amal et al., 2018, Tong et al., 2017). Základní metodou je *náhodná mutace* (*random mutation*), při které je náhodně vybraný gen zaměněn za jiný gen náležící do rozsahu populace. Poměrně značně využívaná je *mutace výměnou*, která spočívá v náhodném výběrů dvou genů a následné výměně jejich pozic v chromozomu. Tato metoda se v literatuře vyskytuje pod více názvy, například Alipour et al. (2018) ji označuje jako *exchanging mutation*, v Notte et al. (2016), Neungmatcha et al. (2013) a Amal et al. (2018) je označena jako *swap/swapping mutation* a Kurnia et al. (2018) se zmiňuje o *position based mutation*. Dále se také můžeme setkat s *aritmetickou mutací* (Nazarahari et al., 2019) a *uniformní mutací* (Assaf et Saleh, 2017, Yi et al., 2016). Některé studie využívají i více metod mutací, například Kurnia et al. (2018) zařadil do genetického algoritmu binární mutaci a mutaci výměnou. Gracia et al. (2013) implementoval do svého algoritmu tři mutační metody, mutaci výměnou, *2-Opt movement* mutaci a *posuvnou mutaci*.



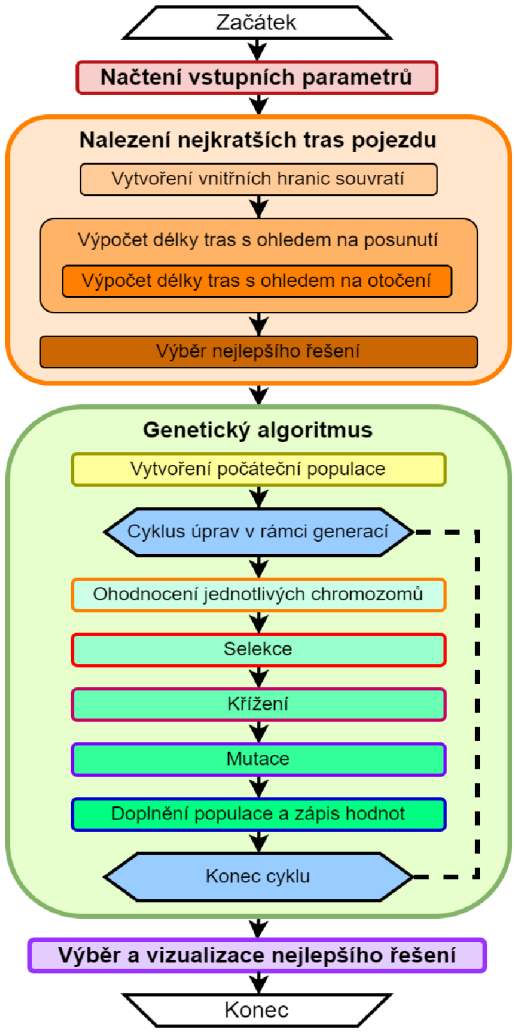
Obrázek 1.10: Znázornění vybraných metod mutace, upraveno z Gracia et al. (2013)

2 Cíl práce

Hlavním cílem dizertační práce je navrhnout a rozpracovat matematický model trajektorie pohybu zemědělské techniky po pozemku zaměřený na nalezení optimálních pojezdových tras aplikací genetických algoritmů, přičemž důraz bude kladen především na otáčení techniky v oblasti souvratí. V rámci práce bude provedena numerická realizace modelu pro zvolené vstupní parametry, přičemž na základě vygenerovaných dat dojde k porovnání implementovaných metod genetických algoritmů. Obsažen bude také přehled mapující využití metaheuristických přístupů při činnosti zemědělských strojů v systémech precizního zemědělství, a to především se zřetelem na využití genetických algoritmů.

3 Algoritmus pro optimalizaci pohybu zemědělské techniky po pozemku s implementací GA

Struktura algoritmu se skládá ze tří hlavních částí, které se rozvětvují do dílčích skriptů jednotlivých funkcí. První část je zaměřena na zpracování vstupních bodů ohraničení pozemku, vytvoření optimálních souvratí s ohledem na daný poloměr otáčení, a především nalezení nejkratších pojezdových tras po pozemku. V části druhé podstupují prvky nejlepšího vygenerovaného řešení proces zpracování pomocí genetických algoritmů pro získání optimální kombinace průjezdů dílčími trasami za účelem snížení pojezdové vzdálenosti při otáčkách na souvratích, zahrnující hodnotící funkci fitness, četné metody selekce i vhodné postupy křížení a mutace nově vzniklých kombinací, z nichž je ta, která vykazuje nejlepší výsledky, v poslední části vizualizována. Na obrázku 3.1 je ukázáno stručné schéma jednotlivých částí, které jsou dále rozpracovány.

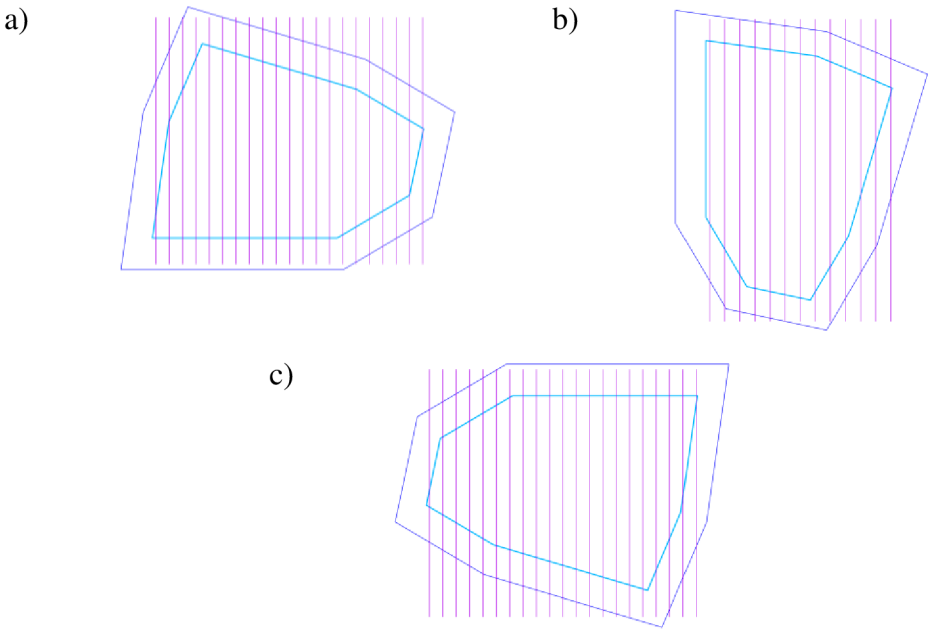


Obrázek 3.1: Schéma algoritmu pro optimalizaci pohybu zemědělské techniky po pozemku

K vstupním parametrům algoritmu patří také záběr operačních prvků vybrané techniky, počet generací, po které bude genetický algoritmus probíhat, počet chromozomů v jedné populaci, tedy počet možných řešení zpracovaných během každé generace, a pravděpodobnost křížení a mutace určující průběh těchto operací. Aplikované algoritmy popisované v rámci následujících částí jsou napsány v programovacím jazyku MATLAB, uveden je i zdrojový kód včetně stručných popisků vybraných řádků pro lepší srozumitelnost.

3.1 Funkce pro výpočet nejkratších pojezdových tras

Implementovaná funkce slouží k nalezení rovnoběžných dílčích tras pojezdu ve vnitřní části pozemku s účelem snížit celkovou vzdálenost pohybu zemědělské techniky simulací rozličných variant orientace trajektorií vůči pozemku při změnách úhlu otočení v rozmezí 0° až 180°, zároveň dochází také k postupnému posunutí drah pojezdu o desetinu záběru v rozsahu 0 až $\frac{9}{10}$ záběru. Pro zjednodušení modelace zvolené situace jsou rovnoběžné úsečky představující dílčí trasy vždy ve vertikální poloze, tedy jejich oba krajní body mají stejnou souřadnici x, a rotovány jsou body polygonu vymezujícího vnitřní část pozemku. Na obrázku 3.2 je znázorněna variabilní orientace trajektorií, na kterých leží dílčí trasy, vůči pozemku při rotaci.



Obrázek 3.2: Ukázka variabilní orientace trajektorií vůči pozemku při rotaci polygonu o a) 0°, b) 90°, c) 180°

Vstupní parametry tvoří proměnné x_s a y_s, což jsou souřadnice x a y bodů vytyčujících vnější hranici pozemku, pracovní záběr z_{ab} a poloměr otáčení r_{ot} vybrané techniky. Výstupem jsou souřadnice mezních bodů dílčích pojezdových tras s nejmenší celkovou délkou $Pmin$, střed otočení S_o a úhel otočení mu. V rámci algoritmu je nejprve proveden výpočet šířky souvratí, následně jsou zjištěny souřadnice středu otočení pozemku v podobě centroidu, tedy

$$S_{o_x} = \sum_{i=1}^{N} x_{s_i} \Big/ N, \qquad S_{o_y} = \sum_{i=1}^{N} y_{s_i} \Big/ N, \tag{3.1}$$

kde N je počet zadaných bodů hranice pozemku, poté je funkcí *polyshape* vytvořen polygon vymezující pozemek, na jeho základě dochází funkcí *polybuffer* k vytvoření polygonu vnitřní části pozemku vytyčujícího oblast souvratě, jež odpovídá vnitřnímu odsazení polygonu pozemku o šířku souvratí. Proměnné x_{s1} a y_{s1} obsahují informace o souřadnicích bodů vnitřního polygonu, které jsou předmětem dalšího zpracování dvou *for* cyklů, kdy během prvního dochází k posunu tras o $\frac{z_{ab}}{10}$, v rámci vnořeného druhého *for* cyklu je měněn úhel otočení pozemku, následně je realizováno otočení bodů vnitřního polygonu a aplikací funkce *f_pruseciky* vygenerovány krajní body dílčích tras pojezdu a v dalším vnořeném *for* cyklu jsou vypočítány jejich vzdálenosti, tedy délky jednotlivých trajektorií, jejichž suma je následně, již mimo tento cyklus, zaznamenána. Výsledkem této fáze je matice celkových délek o rozměrech 10×181, kde sloupce reprezentují jednotlivé úhly otočení a řádky posunutí, přičemž index nejnižší hodnoty matice lze zjistit dvojnásobným použitím funkce min, avšak určen je pouze index sloupce, pro index řádku je třeba matici transponovat. Na základě zjištěných indexů jsou stanoveny hodnoty úhlu i posunutí, při kterých byla minimální hodnota získána, a nakonec dochází k vygenerování souřadnic finálních bodů průjezdu $Pmin$.

Ukázka zdrojového kódu 3.1: Funkce pro výpočet nejkratších pojezdových tras

```
1    function [Pmin,S_o,mu]=f_So_nt(x_s,y_s,z_ab,r_ot)
2
3      %výpočet šířky souvratí
     souv=ceil(2.3*r_ot/z_ab)*z_ab;
4      %výpočet souřadnic středu otočení pozemku
     S_o=[sum(x_s)/length(x_s),sum(y_s)/length(y_s)];
```

```
5       %vygenerování polygonu vnějších hranic pozemku
    pol=polyshape(x_s,y_s);
6       % vygenerování polygonu vnitřní části pozemku
    polin=polybuffer(pol, -souv);
7       %získání souřadnic bodů polygonu vnitřní části pozemku
    x_s1=polin.Vertices(:,1);
8   y_s1=polin.Vertices(:,2);
9   D1=[];
10      %velikost posunutí tras pojezdu na ose x vůči základnímu sestavení, kdy
        má první trasa pojezdu souřadnici x o velikosti min(y_s1)+z_ab/2
    for j=0:z_ab/10:9/10*z_ab
11      D01=[];
12          %úhel otočení pozemku
        for i=0:1:180
13              %vygenerování matic bodů pozemku a souvrati otočených o daný
                úhel
            B_o1=f_otoceni(x_s1,y_s1,S_o,i);
14              %nalezení průsečíků pojezdových tras a souvratí
            P=f_pruseciky (B_o1(1,:),B_o1(2,:),z_ab,j);
15          d=[];
16          for m=1:1:length(P)/2
17                  %výpočet délky dílčích tras pojezdu
                d0=sqrt((P(1,2*m-1)-P(1,2*m))^2+(P(2,2*m-
                1)-P(2,2*m))^2);
18              d=[d,d0];
19          end
20              %záznam sumy délek dílčích tras pro každý úhel otočení ze sta-
                noveného intervalu
            D01=[D01,sum(d)];
21      end
22          %záznam celkové délky tras pro jednotlivé hodnoty posunutí ze stano-
            veného intervalu (skupina všech řešení)
        D1=[D1;D01];
23  end
24      %získání indexu sloupce, kde se nachází nejlepší řešení (nej-
        kratší pojezdové trasy)
    [~,por1]=min(min(D1));
25      %získání indexu sloupce, kde se nachází nejlepší řešení
    [~,por2]=min(min(D1'));
26      %seznam všech úhlů ze zkoumaného intervalu
```

```
uhel=0:1:180;
27    %seznam všech hodnot posunutí ze zkoumaného intervalu
posun=0:z_ab/10:9/10*z_ab;
28    %nalezení úhlu otočení pozemku, při kterém byly nalezeny nejkratší trasy
      pojezdu
mu=uhel(por1);
29    %nalezení hodnoty posunutí, při kterém byly nalezeny nejkratší trasy po-
      jezdu
mp=posun(por2);
30    %otočení bodů pozemku a souvrati o úhel nejlepšího řešení
[BB,BB1]=f_otoceni(x_s,y_s,x_s1,y_s1,S_o,mu);
31    %vygenerování mezních bodů pojezdových tras pro parametry nejlepšího ře-
      šení
Pmin=f_pruseciky (BB1(1,:),BB1(2,:),z_ab,mp);
```

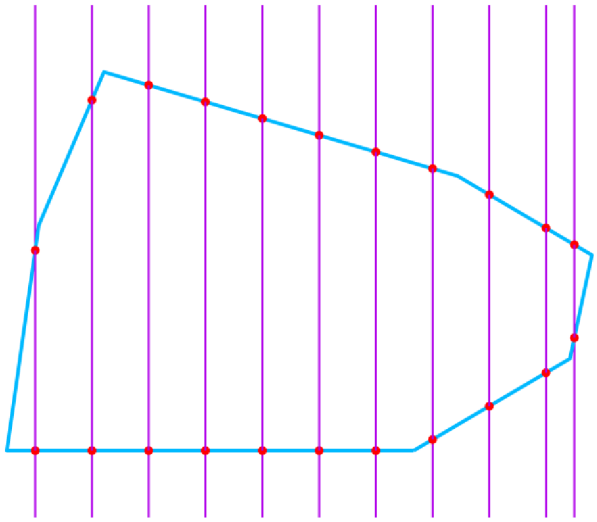
3.2 Funkce pro výpočet mezních bodů dílčích pojezdových tras

Mezní body dílčích tras ve vnitřní části pozemku jsou jejich vstupní a výstupní místa, kde komplexní trajektorie přechází z přímočarého pohybu do otáčení či naopak z otočky do přímé dráhy pojezdu. Obecně lze tyto body definovat jako průsečíky rovnoběžných přímek, jichž jsou dílčí trasy součástí, a polygonu vnitřní části pozemku, avšak pro zajištění kompletního pokrytí polygonu je třeba jejich polohu dále náležitě upravit. Vstupní parametry tvoří x_{s1} a y_{s1}, tedy souřadnice x a y hranic vnitřní části pozemku, záběr techniky z_{ab} a posunutí po, výstupem je množina mezních bodů P. Po vytvoření polygonu a nalezení jeho mezních hodnot x_{min} a x_{max} na ose x je vygenerována skupina souřadnic x dílčích tras v podobě posloupnosti se zápisem

$$a_{n+1} = a_n + z_{ab};\ a_1 = x_{min} + \frac{z_{ab}}{2}, \tag{3.2}$$

dokud $a_{n+1} < x_{max}$, následně je ke každému členu přičtena hodnota posunutí. Poté je ověřeno, zda po posunutí není rozdíl $x_{min} - a_1$ nebo $x_{max} - a_{n+1}$ větší než $\frac{z_{ab}}{2}$, pokud ano, tak je v prvním případě přidán na první místo skupiny prvek $a_0 = a_1 - \frac{z_{ab}}{2}$, v druhé situaci je na konec skupiny zařazen prvek $a_{n+2} = a_{n+1} + \frac{z_{ab}}{2}$. V případě, že by po posunutí byl člen a_{n+1} větší než x_{max}, je ze skupiny odstraněn. Zmíněný postup má zabezpečit dosah záběru pracovního nástroje po celé vnitřní části pozemku. V dalším kroku jsou během for cyklu vytvořeny funkcí *intersect* průsečíky polygonu a úseček představující pojezdové trasy, viz obrázek 3.3, přičemž může nastav stav, kdy v konkávní oblasti pozemku jej úsečka protíná ve více částech a mezi souřadnicemi

průsečíků budou hodnoty NaN (Not a Number). V této situaci dochází k výběru všech prvků nerovnajících se NaN, jejich seřazení do matice o velikosti $n \times 1$, kde n je jejich počet, a úprava matice do tvaru $\frac{n}{2} \times 2$, kdy v prvním sloupci jsou zaneseny souřadnice x a v druhém souřadnice y, dále jsou získané body setříděny dle souřadnice x vzestupně a výsledná matice je transponována.



Obrázek 3.3: Znázornění průsečíků polygonu vnitřní části pozemku a úseček představující trajektorie pojezdových tras

Ukázka zdrojového kódu 3.2: Funkce pro výpočet mezních bodů dílčích pojezdových tras – část 1

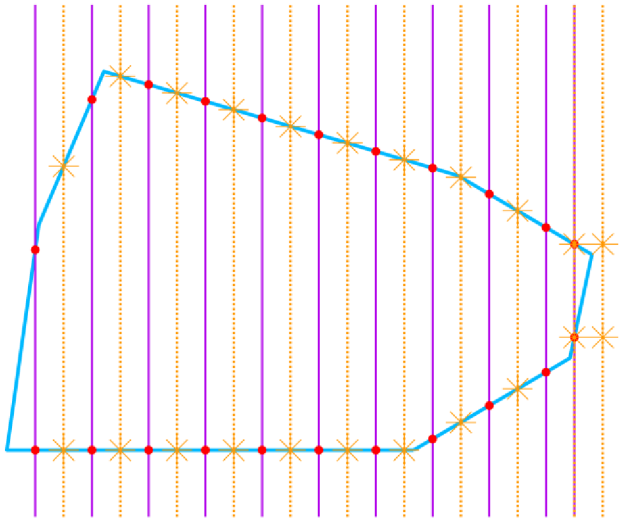
```
1   function [P]=f_pruseciky2_3(x_s1,y_s1,z_ab,po)
2
3   polin=polyshape(x_s1,y_s1);
6   x_max=max(x_s1);
7   x_min=min(x_s1);
8    %list souřadnic x dílčích pojezdových tras
    poc=x_min+z_ab/2:z_ab:x_max;
9    %list souřadnic x dílčích pojezdových tras posunuté o danou hodnotu "po"
    poc=poc+po;
10   %pokud je rozdíl mezi souřadnicí x bodu náležícího dílčí trase pojezdu
     nacházející se první zleva a nejmenší souřadnicí x ze všech bodů hranice
     vnitřní části pozemku větší než polovina záběru
    if poc(1,1)-xmin>z_ab/2
11        %k listu poc je před první souřadnici x přidána souřadnice x o z_ab/2
          menší
        poc=[poc(1,1)-z_ab/2,poc];
12  end
```

```
13      %pokud je souřadnice x dílčí trasy, která je první zprava, větší než
        největší souřadnice x ze všech bodů hranice vnitřní části pozemku, tak je
        třeba tuto souřadnici z listu poc odstranit
    if poc(1,length(poc))>xmax
14          poc=poc(1:length(poc)-1);
15      end
16      %pokud je rozdíl mezi souřadnicí x bodu náležícího dílčí trase pojezdu
        nacházející se první zprava a největší souřadnicí x ze všech bodů hranice
        vnitřní části pozemku větší než polovina záběru
    if x_max-poc(1,length(poc))>z_ab/2
17          %k listu poc je za poslední souřadnici x přidána souřadnice x
            o z_ab/2 větší
            poc=[poc,poc(1,length(poc))+z_ab/2];
18      end
19      inter0=[];
20      %nalezení průsečíků dílčích tras pojezdu s polygonem vnitřní části pozemku
    for i0=1:length(poc)
21          %vygenerování průsečíků polygonu vnitřní části pozemku a dílčích
            pojezdových tras
            inter00=intersect(polin,[poc(i0),min(y_s1)-
            10;poc(i0),max(y_s1)+10]);
22          %pokud je v nalezené matici hodnota NaN, znamená to, že úsečka, na
            které leží pojezdová trasa protíná pozemek ve více bodech
            if sum(sum(isnan(inter00))) > 0
23              %nalezení prvků nemající hodnotu NaN, dochází k seřazení prvků
                do matice velikosti (n, 1), kde n je počet prvků non NaN
                ii0=inter00(~isnan(inter00));
24              %převedení matice do původního tvaru, kde ve sloupci 1 jsou
                souřadnice x a ve sloupci 2 souřadnice y
                inter00=reshape(ii0,[length(ii0)/2,2]);
25          end
26          inter0=[inter0;inter00];
27      end
28      %seřazení hodnot vzestupně dle velikosti podle prvního sloupce a následná
        transpozice matice
    P=sortrows(inter0)';
```

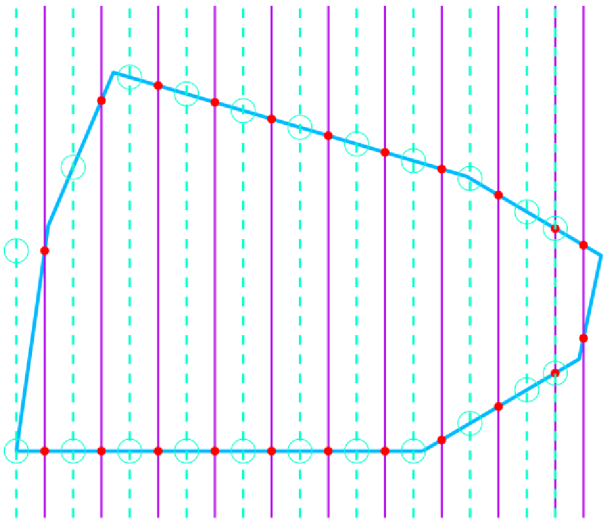
⋮

Další úprava je zaměřena na vygenerování průsečíků vnitřní části pozemku s pomocnými rovnoběžkami dílčích pojezdových tras, jenž se liší v souřadnici x o $\pm\frac{z_{ab}}{2}$, viz obrázky 3.4 a 3.5, které dále poslouží úpravě pozice stávajících průsečíků. Pokud

pro některé rovnoběžky neprotínají pozemek, standardně se jedná o rovnoběžku posunutou o $-\frac{z_{ab}}{2}$ od levé krajní trasy a rovnoběžku posunutou o $\frac{z_{ab}}{2}$ od pravé krajní trasy, tak jsou body simulující průsečíky P_{Ls}, P_{Lh}, P_{Ps}, P_{Ph} vytvořeny posunutím průsečíků trasy T_s, T_h, což jsou body průjezdu, s polygonem o $-\frac{z_{ab}}{2}$, případně o $\frac{z_{ab}}{2}$, tedy $P_{Ls} = [T_{s_y} - \frac{z_{ab}}{2}; T_{s_y}]$, $P_{Lh} = [T_{h_x} - \frac{z_{ab}}{2}; T_{h_y}]$, $P_{Ps} = [T_{s_x} + \frac{z_{ab}}{2}; T_{s_y}]$, $P_{Ph} = [T_{h_x} + \frac{z_{ab}}{2}; T_{h_y}]$. Pokud je počet nalezených průsečíků větší než dva, tak dochází k výběru té úsečky, jejíž spodní bod má menší vzdálenost ke spodnímu bodu příslušné trasy. Když je zkoumaná část polygonu konkávní, tak může také nastat situace, že rovnoběžka protíná polygon v jiné oblasti, která sice polohou vůči ose x odpovídá dané trase, ale poloha k ose y je značné rozdílná, a tak dochází ještě k ověření adekvátnosti polohy srovnáním polohy horního bodu trasy T_h se spodním bodem rovnoběžné úsečky P_s a spodního bodu T_s trasy s horním bodem rovnoběžky P_h. Pokud má horní bod trasy menší souřadnici y než spodní bod rovnoběžky či spodní bod souřadnici y větší než horní bod rovnoběžky, tedy $T_{h_y} < P_{s_y}$ nebo $T_{s_y} > P_{h_y}$, tak jsou vytvořeny nové body rovnoběžek posunutím bodů trasy o $-\frac{z_{ab}}{2}$, případně $\frac{z_{ab}}{2}$.



Obrázek 3.4: Znázornění rovnoběžných úseček vytvořených posunutím pojezdových tras o z_ab/2 a jejich průsečíků s polygonem



Obrázek 3.5: Znázornění rovnoběžných úseček vytvořených posunutím pojezdových tras o -z_ab/2 a jejich průsečíků s polygonem

Ukázka zdrojového kódu 3.3: Funkce pro výpočet mezních bodů dílčích pojezdových tras – část 2

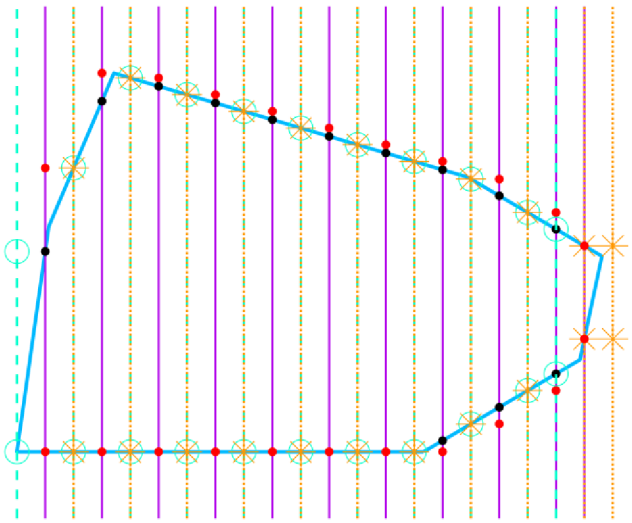
```
                                    ⋮
29      inter1=[];
30       %nalezení průsečíků vnitřní části pozemku s rovnoběžkami dílčích pojez-
         dových tras jenž se liší v souřadnici x o +-záběr/2,
        for i1=1:length(P)/2
31          inter01=intersect(polin,[P(1,i1*2-
            1)+z_ab/2,min(y_s1)-
            10;P(1,i1*2)+z_ab/2,max(y_s1)+10]);
32          inter02=intersect(polin,[P(1,i1*2-1)-
            z_ab/2,min(y_s1)-10;P(1,i1*2)-
            z_ab/2,max(y_s1)+10]);
33           %pokud nebyly nalezeny žádné průsečíky, tak bude rovnoběžná úsečka
             stejná jako pojezdová trasa, pouze o z_ab/2 posunutá doprava
            if isempty(inter01)
34              inter01=[P(1,i1*2-1)+z_ab/2,P(2,i1*2-
                1);P(1,i1*2)+z_ab/2,P(2,i1*2)];
35          end
36           %pokud nebyly nalezeny žádné průsečíky, tak bude rovnoběžná úsečka
             stejná jako pojezdová trasa, pouze o z_ab/2 posunutá doleva
            if isempty(inter02)
37              inter02=[P(1,i1*2-1)-z_ab/2,P(2,i1*2-
                1);P(1,i1*2)-z_ab/2,P(2,i1*2)];
38          end
39          if sum(sum(isnan(inter01)))>0
40              ii1=inter01(~isnan(inter01));
```

```
41              inter01=reshape(ii1,[length(ii1)/2,2]);
42          end
43          if sum(sum(isnan(inter02)))>0
44              ii1=inter02(~isnan(inter02));
45              inter02=reshape(ii1,[length(ii1)/2,2]);
46          end
47          %pokud je počet nalezených průsečíků větší než dva, tak průnikem
            vznikají dvě úsečky, je tedy třeba přiřadit k příslušné pojezdové
            trase vhodnou rovnoběžnou úsečku ze dvou možných
            if size(inter01,1) > 2
48              %pokud je absolutní hodnota vzdálenosti mezi spodním bodem
                trasy a spodním bodem první úsečky menší vzdálenost mezi ten-
                týž bodem a spodním bodem druhé úsečky, budou přiřazeny body
                úsečky 1, jinak body úsečky 2
                if abs(P(2,i1*2-1)-inter01(1,2))<abs(P(2,i1*2-
                1)-inter01(3,2))
49                  inter01=inter01(1:2,:);
50              else
51                  inter01=inter01(3:4,:);
52              end
53          end
54          if size(inter02, 1)>2
55              if abs(P(2,i1*2-1)-inter02(1,2))<abs(P(2,i1*2-
                1)-inter02(3,2))
56                  inter02=inter02(1:2,:);
57              else
58                  inter02=inter02(3:4,:);
59              end
60          end
61          inter1=[inter1;inter01,inter02];
62      end
63      P1=sortrows(inter1)';
64      for q=1:length(P1)/2
65          %pokud má úsečka rovnoběžná s dílčí pojezdovou trasou zprava spodní
            bod položený výše než horní bod pojezdové trasy nebo horní bod níže
            než spodní bod pojezdové trasy, tak jsou souřadnice y jejích bodů
            nahrazeny příslušnými souřadnicemi y bodů dílčí trasy pojezdy
            if P1(2,2*q-1)>P(2,2*q) | P1(2,2*q)<P(2,2*q-1)
66              P1(2,2*q-1)=P(2,2*q-1);
67              P1(2,2*q)=P(2,2*q);
68          end
```

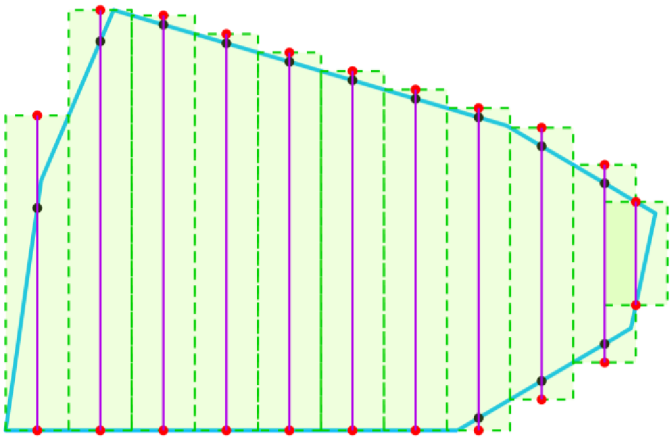
```
69          %stejná podmínka jako výše, pouze se týká rovnoběžné úsečky nacha-
            zející se nalevo
           if P1(4,2*q-1)>P(2,2*q) | P1(4,2*q)<P(2,2*q-1)
70             P1(4,2*q-1)=P(2,2*q-1);
71             P1(4,2*q)=P(2,2*q);
72         end
73     end
```

⋮

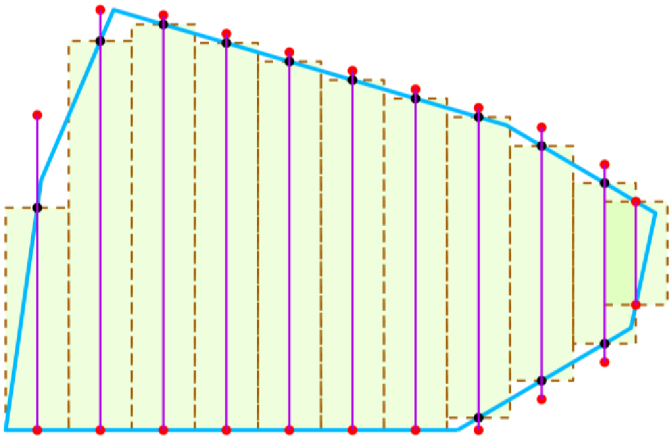
Nakonec dochází k postupnému porovnání souřadnic y spodních bodů T_{S_y} jednotlivých pojezdových drah se spodními body příslušných pomocných rovnoběžek P_{Ls_y}, P_{P_y} a body polygonu $B_1, \dots, B_m$ se souřadnicemi x v intervalu $\langle T_{s_x} - \frac{z_{ab}}{2}; T_{s_x} + \frac{z_{ab}}{2} \rangle$, pokud takové body existují a jsou nalezeny pomocí funkce *find*. Když je jejich hodnota větší než minimální hodnota ze skupiny srovnávaných bodů $P_{min} = \min(T_{s_y}, P_{Ls_y}, P_{P_y}, B_{1_y}, \dots, B_{m_y})$, tak je hodnotou P_{min} nahrazena. Úprava se stejnou analogií je aplikována i na horní body, avšak zde se porovnává s maximální hodnotou souřadnic y vybraných bodů, viz obrázek 3.6. Tento proces je proveden za účelem zajištění pokrytí celé vnitřní částí pozemku pracovním nástrojem, jak je znázorněno na obrázku 3.7, protože pokud by byly využity průsečíky vygenerované v první části algoritmu, tak by mohlo docházet k vynechání úseků oblastí na krajních pomezí zón záběru dílčích tras, viz obrázek 3.8.



Obrázek 3.6: Znázornění vzniku finálních bodů průjezdu úpravou polohy průsečíků polygonu vnitřní části pozemku a pojezdových tras



Obrázek 3.7: Ukázka pokrytí pozemku pracovním nástrojem při průjezdu mezních bodů s upravenou polohou



Obrázek 3.8: Ukázka pokrytí pozemku pracovním nástrojem při průjezdu mezních bodů bez upravené polohy

Ukázka zdrojového kódu 3.4: Funkce pro výpočet mezních bodů dílčích pojezdových tras – část 3

⋮

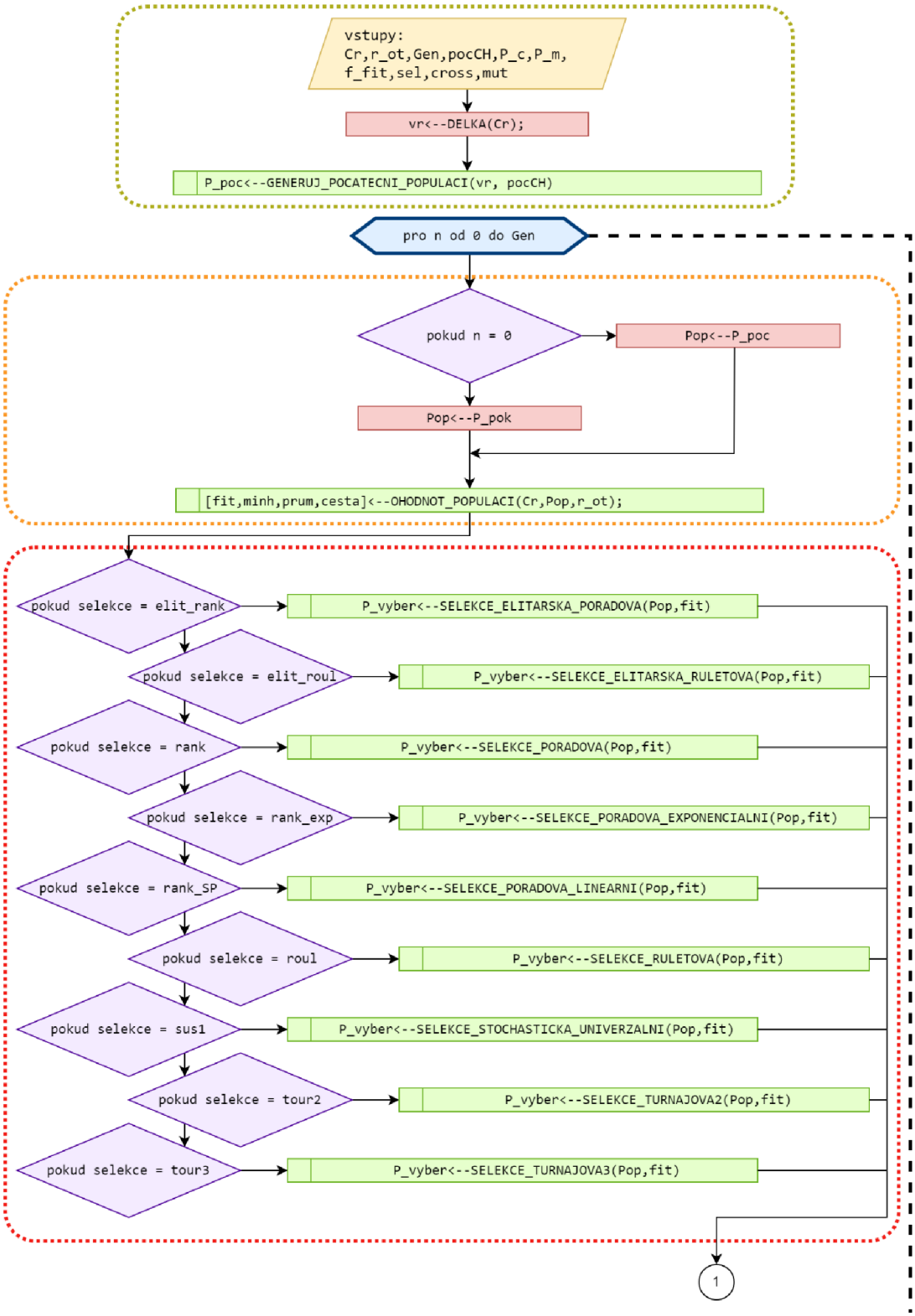
```
74    for m=1:length(P1)/2
75        %úprava proběhne pro trasy 1 až n, kromě situace, kdy bude vybrána
          trasa 1 (krajní trasa zleva) a zároveň její vzdálenost od trasy 2
          bude menší než záběr nebo když bude vybrána trasa n (krajní trasa
          zprava) a zároveň její vzdálenost od trasy n-1 bude menší než zá-
          běr

        if ~((m == 1 & (P(1,3)-P(1,1)<zaber)) | (m==length(P1)/2
        & (P(1,length(P1)-1)-P(1,length(P1)-3)<zaber)))
76          %seřazení souřadnic x rovnoběžných úseček z oblasti -z_ab/2 až
            z_ab/2

          a=sort([P1(1,m*2-1),P1(3,m*2-1)]);
77          %ověření, zda se v okolí -z_ab/2 až z_ab/2 dílčí pojezdové
            trasy nachází bod vnitřní hranice pozemku
```

```
                op=find(x_s1<a(2)&x_s1>a(1));
78              %pokud existují body vnitřní hranice pozemku, jejichž souřadnice x
                odpovídají podmínce výše, tak jsou jejich minimální/maximální sou-
                řadnice y zařazeny do výběru pro přiřazení minima/maxima k souřadni-
                cím y spodních/horních bodů dílčí trasy pojezdu
                if ~isempty(op)
79                  ymin=min([P1(2,m*2-1),P1(4,m*2-1),
                    min(y_s1(op))]);
80                  ymax=max([P1(2,m*2),P1(4,m*2),max(y_s1(op))]);
81              else
82                  %nalezení minima souřadnic y spodních bodů úseček rovnoběžných
                    s dílčí trasou pojezdu nacházejících se v jejím okolí -z_ab/2
                    až z_ab/2
                    ymin=min([P1(2 m*2-1),P1(4,m*2-1)]);
83                  %nalezení maxima souřadnic y horních bodů úseček rovnoběžných
                    s dílčí trasou pojezdu nacházejících se v jejím okolí -z_ab/2
                    až z_ab/2
                    ymax=max([P1(2,m*2),P1(4,m*2)]);
84              end
85              %pokud má spodní bod trasy větší souřadnici y než je nalezené mi-
                nimum, tak je hodnota této souřadnice nahrazena právě vypočítaným
                minimem
                if P(2,m*2-1)>ymin
86                  P(2,m*2-1)=ymin;
87              end
88              %pokud má horní bod trasy menší souřadnici y než je nalezené ma-
                ximum, tak je hodnota této souřadnice nahrazena právě vypočítaným
                maximem
                if P(2,m*2)<ymax
89                  P(2,m*2)=ymax;
90              end
91          end
92      end
```

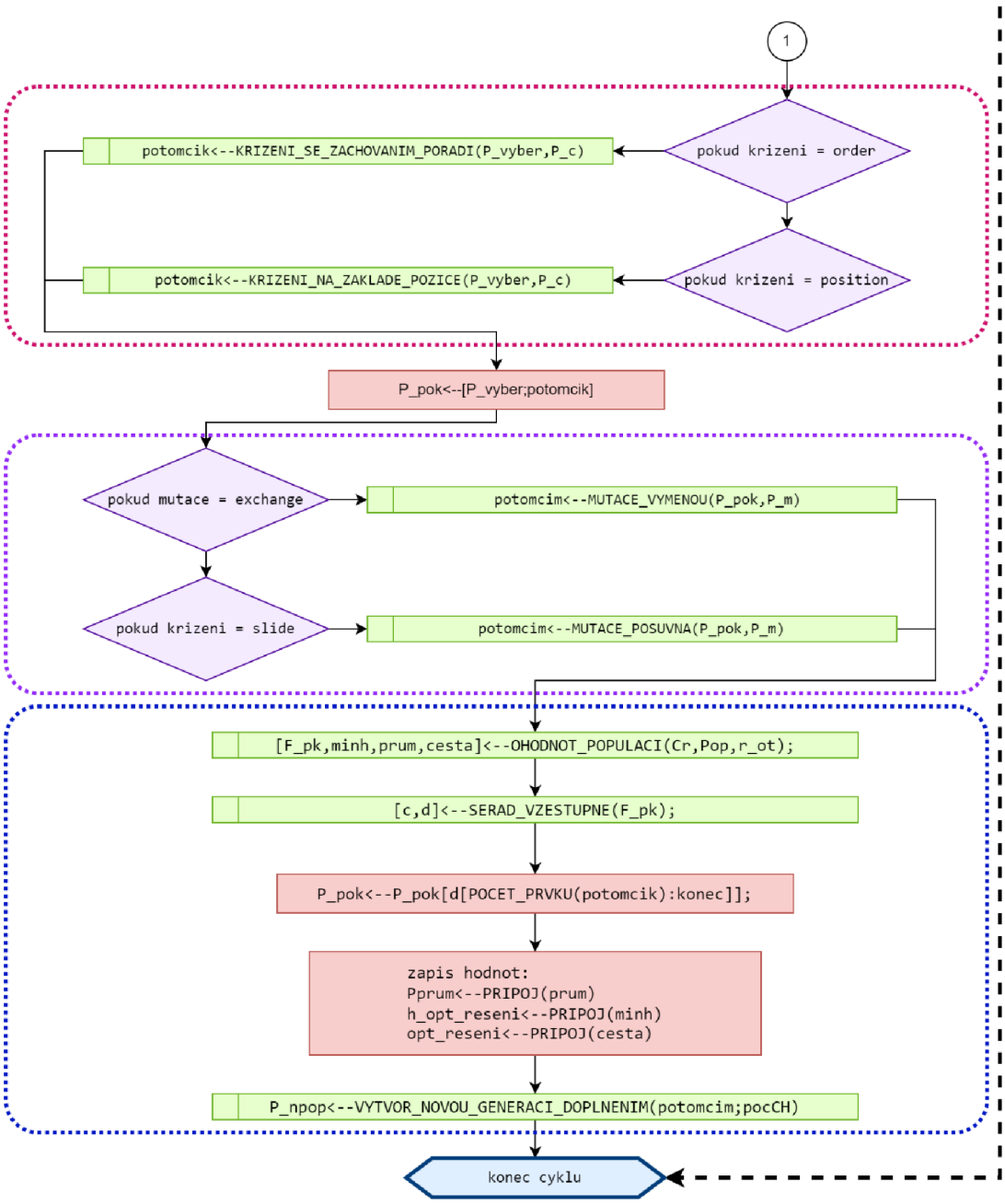
3.3 Funkce genetického algoritmu

Vstupy tvoří pole dvojic mezních bodů dílčích rovnoběžných tras pojezdu po pozemku, Cr, o velikosti $1 \times počet\_bodů/2$, přičemž každá buňka je matice o rozměru 2×2 nesoucí v horním řádku informace o souřadnicích x a spodním řádku o souřadnicích y daných bodů, poloměr otáčení, r_ot, počet epoch či tzv. generací, Gen, počet vstupních řešení průjezdů dílčích tras představujících soubor možných řešení tzv. po-

pulace nebo také chromozom, $pocCH$, pravděpodobnost křížení, P_c, pravděpodobnost mutace, P_m, typ selekce, sel, typ křížení, $cross$, a typ mutace, mut. K výstupům se řadí list obsahující délky nejkratších tras pojezdu při otáčení z celé populace získané během každé epochy, h_opt, list bodů dílčích tras seřazených v pořadí, ve kterém dosahuje pojezd při otáčení nejmenší délky z celé populace, v němž jsou zahrnuta nejlepší řešení pro každou epochu, opt, a list zaznamenávající průměrné hodnoty délek všech řešení každé jednotlivé epochy, $Pprum$. Na počátku algoritmu je nejprve pomocí inicializační funkce vytvořena skupina prvotních řešení o stanoveném počtu, kdy délka každého řešení odpovídá počtu prvků vstupního pole bodů. Hlavní část se skládá z *for* cyklu opakovaného dle zvoleného počtu epoch, z nichž v každé proběhne generování a úprava prvků skupiny možných řešení, přičemž v generaci 0 je zpracována populace vzniklá při inicializaci, v dalších generacích je operováno s populací vzniklou spojením poloviny nejlepších výstupu z generace $n - 1$ a nově vygenerovaných řešení, tzv. jedinců. Nejprve dochází k ohodnocení vstupních řešení hodnotící funkcí *f_GAp02_fit*, následně je z nich aplikací selekční funkce vybrána polovina pro další úpravy. Na obrázcích 3.9 a 3.10 je pro lepší orientaci znázorněn vývojový diagram implementovaného genetického algoritmu s popisky ve stylu pseudokódu, v další části již navazuje okomentovaný zdrojový kód.



Obrázek 3.9: Vývojový diagram implementovaného genetického algoritmu – část 1



Obrázek 3.10: Vývojový diagram implementovaného genetického algoritmu – část 2

V algoritmu je implementováno devět typů selekce:

a) elitářský výběr s pořadovou selekcí,

b) elitářský výběr s ruletovou selekcí,

c) pořadová selekce,

d) pořadová exponenciální,

e) pořadová lineární,

f) ruletová,

g) stochastická univerzální metoda,

h) turnajová s výběrem dvou jedinců,

i) turnajová s výběrem dvou jedinců.

Vybraní jedinci jsou zpracování metodou křížení, kdy dochází k vytvoření dvou nových kombinací tzv. potomků ze dvou původních, rodičů. Vzhledem k tomu, že se v rámci celé trasy nesmí body opakovat vícekrát, byly vybrány pouze dva typy křížení, v rámci, kterých riziko takové situace nehrozí, a to křížení se zachováním pořadí a křížení na základě pozice. Po této operaci jsou vzniklí potomci sloučeni s populací získanou selekcí a celý soubor je podstoupen mutaci, u níž pro výběr metod platí stejná podmínka jako u křížení, implementována je tedy mutace výměnou a posuvná mutace. V dalším kroku je populace znovu ohodnocena a jako základ pro novou populaci je vybrána polovina všech jedinců s nejlepším hodnocením, která je poté pomocí funkce *f_GAp02_fit* doplněna o nově vygenerovaná řešení tak, aby nově vzniklá skupina měla zvolený počet řešení.

Ukázka zdrojového kódu 3.5: Funkce genetického algoritmu

```
1  function [h_opt_reseni,opt_re-
   seni,Pprum]=f_GA(Cr,r_ot,Gen,pocCH,P_c,P_m,sel,cross,mut)
2
3  vr=length(Cr);
4  P_poc=f_GAp01_init(vr,pocCH);
5  P_npop=[];
6
7  Pprum=[];
8  h_opt_reseni=[];
9  opt_reseni=[];
10 C_opt=[];
11 for n=0:Gen
12       % v generaci n=0 je zpracován soubor řešení (populace) vzniklý při
         inicializaci, v dalších generacích je operováno s populací vzniklou
         spojením poloviny nejlepších výstupu z generace n-1 a nově vygenerova-
         ných řešení
     if n==0
13       Pop=P_poc
14   else
15       Pop=P_npop
16   end
```

```
17          %ohodnocení populace
        [fit,~,~,~]=f_GAp02_fit(Cr,Pop,r_ot);
18          %výběr řešení postupujících k dalšímu zpracování (chromozomů) aplikací
            vybraného typu selekce
        if strcmp(sel,'elit_rank')
19            P_vyber=f_GAp03_sel_elit_rank1(Pop,fit);
20        elseif strcmp(sel,'elit_roul')
21            P_vyber=f_GAp03_sel_elit_roul1(Pop,fit);
22        elseif strcmp(sel,'rank')
23            P_vyber=f_GAp03_sel_rank1(Pop,fit);
24        elseif strcmp(sel,'rank_exp')
25            P_vyber=f_GAp03_sel_rank_exp1(Pop,fit);
26        elseif strcmp(sel,'rank_SP')
27            P_vyber=f_GAp03_sel_rank_SP1(Pop,fit);
28        elseif strcmp(sel,'roul')
29            P_vyber=f_GAp03_sel_roulette1(Pop,fit);
30        elseif strcmp(sel,'sus')
31            P_vyber=f_GAp03_sel_sus1(Pop,fit);
32        elseif strcmp(sel,'tour2')
33            P_vyber=f_GAp03_sel_tournament2(Pop,fit);
34        elseif strcmp(sel,'tour3')
35            P_vyber=f_GAp03_sel_tournament3(Pop,fit);
36        end
37
38          %vytvoření populace vybraných chromozomů a jejich potomků vzniklých
            křížením se zachováním pořadí či na základě pozice
        if strcmp(cross,'order')
39            potomcik=f_GAp04cross_order(P_vyber,P_c);
40        elseif strcmp(cross,'position')
41            potomcik=f_GAp04cross_pos01(P_vyber,P_c);
42        end
43
44          %sloučení populace vzniklé při křížení se skupinou jejich rodičů
        P_pok=[P_vyber;potomcik];
45          %aplikace vybrané metody mutace s pravděpodobností P_m
        if strcmp(mut,'exchange')
46            [potomcim]=f_GAp05mut_exch(P_pok,P_m);
```

```
47      elseif strcmp(mut,'slide')
48         [potomcim]=f_GAp05mut_slide(P_pok,P_m);
49      end
50       %po novém ohodnocení všech prvků dochází k výběru poloviny všech je-
         dinců s nejlepším hodnocením
        [F_pk,minh,prum,cesta]=f_GAp02_fit(Cr,P_pok,r_ot);
51      [c,d]=sort(F_pk);
52      P_pok=P_pok(d(size(potomcik,1):end),:);
53       %záznam průměrné délky tras pojezdu při otáčení v rámci celé populace
        Pprum=[Pprum;prum];
54       %záznam délky nejkratších tras pojezdu při otáčení z celé populace
        h_opt_reseni=[h_opt_reseni;minh];
55       %záznam pořadí průjezdu bodů tras s nejmenší délkou pojezdu při otáčení
         z celé populace
        opt_reseni=[opt_reseni;cesta];
56       %vytvoření nové generace, kde je populace po aplikaci mutace doplněna o
         nově vygenerovaná řešení tak, aby nově vzniklá skupina měla zvolený po-
         čet řešení
        P_npop=f_GAp06new_gen(potomcim,pocCH);
57      [fit,~,~,~]=f_GAp02_fit(Cr,P_npop,r_ot);
58      n
59  end
```

3.4 Inicializační funkce

První inicializační funkce je použita pouze v první epoše genetického algoritmu a jejím výstupem je počáteční populace, *P_poc*, tedy soubor možných náhodně vygenerovaných řešení, chromozomů, o délce d, která odpovídá počtu dílčích tras pojezdu, počet jednotlivých řešení v celém souboru je dán stanovenou vstupní hodnotou $pocCH$. Druhá funkce je volána na konci každé epochy pro vytvoření nové generace *P_npop*, kde jsou řešení po mutaci, $potomcim$, doplněna o nově vygenerovaná řešení tak, aby nově vzniklá skupina měla zvolený počet řešení $pocCH$. Hlavní část obou funkcí je tvořena *for* cyklem, v rámci kterého vznikají nová řešení. Pro případ, že se vygenerované řešení v souboru již nachází, je zde vložen cyklus *while*, který generuje nová řešení, dokud není získáno řešení odlišené od všech prvků ve skupině. Následně je nové řešení přidáno do stávající skupiny.

Ukázka zdrojového kódu 3.6: Funkce k vytvoření počáteční populace

```
function [P_poc]=f_GAp01_init(d,pocCH)

P_poc=[];
for i=1:pocCH
      %vektor (chromozom) určující náhodné pořadí buněk
    r=randperm(d);
      %pokud se vygenerovaný chromozom již ve skupině nachází, tak je gene-
      rován znovu dokud není od prvků skupiny odlišný
    while i>1 & sum(ismember(P_poc,r,'rows'))==1
        r=randperm(d);
    end
    P_poc=[P_poc;r];
end
```

Ukázka zdrojového kódu 3.7: Funkce k vytvoření populace n+1

```
function [P_npop]=f_GAp06new_gen(potomcim,pocCH)

n=size(potomcim,2);
  %počet nových chromozomů, které je třeba vygenerovat
tnt=pocCH-size(potomcim,1);
pop=[];
for k=1:tnt
    r=randperm(n);
    while i>1 & (sum(ismember(P_poc,r,'rows')) | sum(is-
         member(potomcim,r,'rows'))==1)

        r=randperm(n);
    end
    pop=[pop;r];
end
P_npop=[potomcim;pop];
```

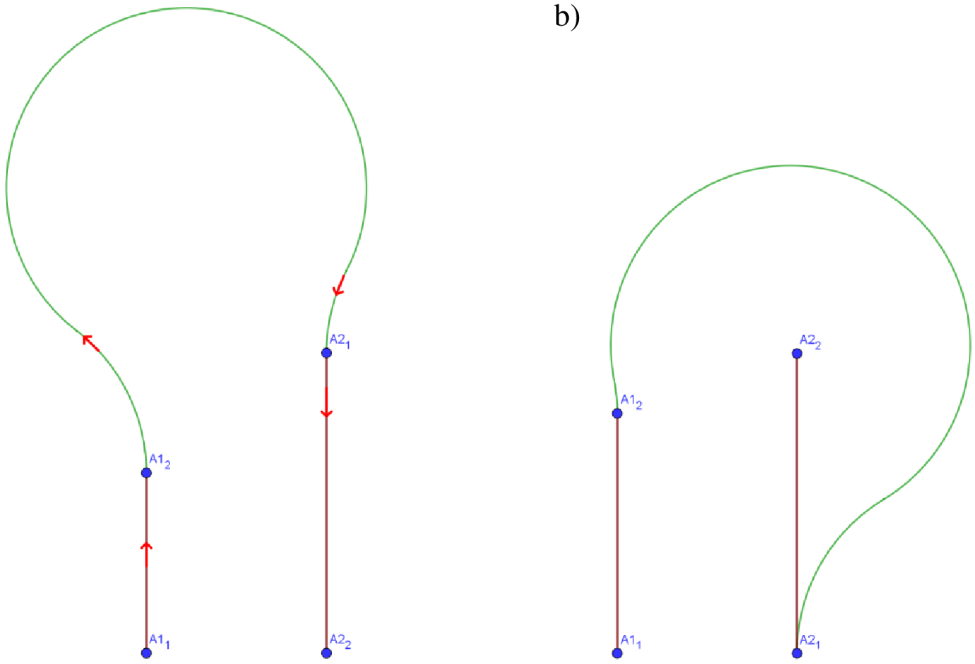
3.5 Funkce fitness k ohodnocení jednotlivých řešení

Tato funkce slouží k ohodnocení jednotlivých řešení ze skupin (chromozomů) vzniklých v rámci každého kola cyklu (generace) algoritmu. Hlavní část funkce se skládá ze dvou for cyklů, kdy v rámci vnějšího cyklu dochází k průchodu všech členů souboru

vstupních řešení, přičemž jsou pokaždé prvky v poli bodů průjezdu seřazeny dle konkrétního řešení, na konci cyklu dochází k záznamu celkové délky pojezdu po souvratích pro každé ze vstupních řešení a zápis jednotlivých souborů buněk bodů průjezdu s novým pořadím. Vnitřní cyklus operuje postupně se všemi dvojicemi bodů průjezdu, jež jsou reprezentovány v každé buňce pole, dle daného pořadí, přičemž dochází k výběru dvou po sobě jdoucích buněk $A1$, $A2$. První bod buňky A1 je brán jako bod vstupu do první z vybraných dílčích tras, druhý bod, bod $A1_2$ je výstupní, tedy počátek otočky, v prvním bodu buňky $A2$, bodu $A2_1$ je konec otočky a zároveň vstup do druhé z vybraných dílčích tras. Ke správnému provedení otočky musí být vstupní i výstupní bod otočky ve shodné poloze vůči zbývajícím krajním bodům vybraných tras, tedy má-li souřadnice y bodu $A1_2$ vetší hodnotu než souřadnice y bodu $A1_1$, musí mít souřadnice y bodu $A2_1$ vetší hodnotu než souřadnice y bodu $A2_2$, tato situace je znázorněna na obrázku 3.11 a). V případě, že má souřadnice y bodu $A1_2$ vetší hodnotu než souřadnice y bodu $A1_1$ a zároveň souřadnice y bodu $A2_1$ menší hodnotu, než souřadnice y bodu $A2_2$, je třeba pořadí bodů v buňce $A2$ převrátit, protože by jinak došlo k vynechání části trasy pojezdu a otočka by zasahovala do vnitřní části pozemku, viz obrázek 3.11 b). Odpovídající podmínka platí i pro situaci, kdy je poloha otočky opačná, tedy mezní body otočky mají souřadnice y s nižší hodnotou než zbylé krajní body vybraných tras.

Dále je zjištěna poloha otočky vůči trasám pojezdu, zda se nachází nad či pod body průjezdu, a následně je pomocí funkce *f_So_ot_vz1* zjištěna délka celé otočky. V rámci algoritmu je proveden součet délek všech otoček provedených během průjezdu pozemku a uložen pro každé zkoumané řešení, zároveň jsou také zapsány jednotlivé soubory buněk bodů průjezdu s upraveným pořadím. Nakonec dochází k ohodnocení každého ze vstupních řešení, výpočtu průměrné vzdálenosti pojezdu po souvratích, nalezení nejlepšího řešení s nejkratší délkou pojezdu v rámci otoček a zápisu bodů trasy průjezdů nejlepšího řešení včetně jejich upraveného pořadí v rámci dílčích tras pojezdu po pozemku.

a) b)



Obrázek 3.11: Vizualizace průjezdu krajních bodů dvou dílčích tras pojezdu

Ukázka zdrojového kódu 3.8: Funkce pro ohodnocení jednotlivých řešení

```
1   function [fit,minh,prum,cesta]=f_GAp02_fit(Cr,P_poc,r_ot)
2
3   d1=[];
4   for i=1:size(P_poc,1)
5       d=[];
6       %seřazení prvků pole dle pořadí z daného prvku vstupního řešení (chro-
        mozomu)
        Cr1=Cr([P_poc(i,:)]);
7       for j=1:size(Cr1,2)-1
8           %výběr dvou sousedních buněk v rámci jedné vrstvy 3D pole (sudá
            a lichá)
            A1=Cr1{:,j};
9           A2=Cr1{:,j+1};
10          %seřazení prvků ve dvou vybraných sousedních buňkách tak, aby po
            položení buněk vedle sebe měli body dotyku vybraných buněk stej-
            nou polohu vůči krajním bodům buněk, tedy aby například krajní
            body byly nahoře a body dotyku dole
            if (A1(2,1)<A1(2,2) & A2(2,1)<A2(2,2)) |
               (A1(2,1)>A1(2,2) & A2(2,1)>A2(2,2))
11              %přehození pořadí bodů v buňce
                A2=[A2(:,2),A2(:,1)];
12              %zápis upraveného pořadí do původního souboru buněk
                Cr1{:,j+1}=A2;
```

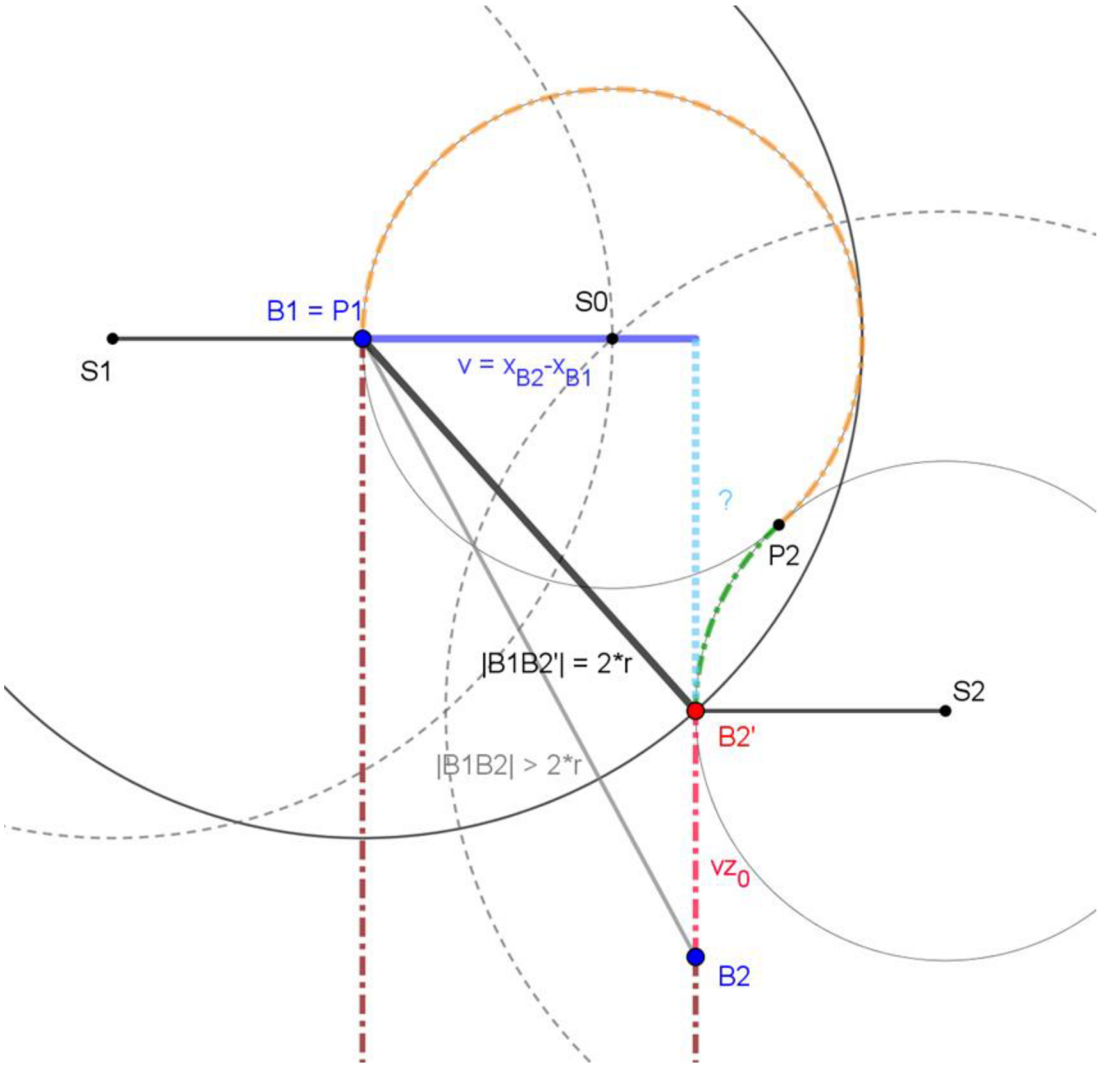
```
13              end
14               %výběr x-souřadnic sousedních bodů vybraných buněk (u 1. buňky
                 ze dvou vybraných je to druhý bod, u 2. buňky první bod)
                xB=[A1(1,2),A2(1,1)];
15               %výběr x-souřadnic sousedních bodů vybraných buněk
                yB=[A1(2,2),A2(2,1)];
16               %ověření, zda je otočka nad či pod body průjezdu
                if A1(2,2)>A1(2,1)
17                  sm=1;
18              else
19                  sm=0;
20              end
21               %výpočet délky trasy otočky
                [vzdalenost]=f_So_ot_vz1(xB,yB,r_ot,sm);
22               %záznam délky každé otočky v rámci konkrétního řešení
                d=[d,vzdalenost];
23          end
24           %záznam celkové délky pojezdu po souvratích pro každé ze vstupních
             řešení (suma délek tras otoček)
            d1=[d1;sum(d)];
25           %zápis jednotlivých souborů buněk s novým pořadím
            Cr2(:,:,i)=Cr1;
26      end
27
28      d1=real(d1);
29       %ohodnocení každého řešení na základě podílu součtu délek všech řešení a
         jednotlivých řešení, tedy čím menší celková délka pojezdu po souvratích, tím
         větší bodová hodnota
        fit=sum(d1)./d1;
30       %výpočet průměrné délky pojezdu v rámci všech vstupních řešení
        prum=sum(d1)/length(d1);
31       %nalezení nejlepšího řešení (s nejkratší délkou pojezdu) a jeho pořadí
        [minh,p]=min(d1);
32      %přiřazení bodů trasy průjezdů nejlepšího řešení včetně jejich upraveného
        pořadí v rámci dílčích tras pojezdu po pozemku
        cesta=Cr2(:,:,p);
```

3.6 Funkce pro výpočet délky otočky

Vstupními parametry této funkce jsou proměnné xB, yB, ve kterých jsou zapsány souřadnice dvou sousedních bodů průjezdu pojezdové trasy, $B1$ a $B2$, tak, že xB obsahuje hodnotu souřadnic na ose x a yB na ose y. Proměnná r_ot je poloměr otáčení techniky a sm vyznačuje, zda je otočka situována nad body průjezdu či pod, tedy zda y-ové složky souřadnic bodů otočky jsou větší než nejmenší y-ové hodnoty vstupních bodů či naopak. Výstupem je délka celé otočky. Po načtení vstupních hodnot jsou nejprve body seřazeny tak, aby bod $B1$ měl menší souřadnici x než bod $B2$, což zjednodušuje správné vypočítání hodnot středů kružnic, jež jsou součástí dráhy otočky, s ohledem na typ možné situace polohy bodů průjezdu v souvislosti s poloměrem otáčení a směrem průjezdu. Následně je vypočítána horizontální vzdálenost v mezi vstupními body, která slouží jako hlavní část podmínky k výpočtu délky otočky. Pokud je menší a zároveň vzdálenost vstupních bodů je větší než dvojnásobek poloměru otáčení, tak je třeba posunout souřadnici y toho bodu, u kterého má tato souřadnice v absolutní hodnotě nižší hodnotu, blíže k souřadnici y druhého bodu tak, aby vzdálenost bodů byla rovna dvojnásobku poloměru otáčení, viz obrázek 3.12. V případě, že je otočka nad body průjezdu, tak je bod situovaný níže posunut nahoru ve směru osy y, v opačném případě zase dolů, velikost tohoto posunu, tedy vzdálenost rovna rozdílu mezi původní a novou hodnotou bodu, musí být započítána do celkové délky otočky. Pokud je vzdálenost $|B1B2| > 2r$, otočka směřuje nad body průjezdu a $y_{B1} > y_{B2}$, je tedy třeba posunout bod $B2$ tak, aby $|B1B2'| = 2r$, přičemž bod B2‘ má souřadnice $[x_{B2}, y_{B2}']$. Novou hodnotu $y_{B2'}$ lze vypočítat jako

$$y_{B2'} = y_{B1} - \sqrt{4r^2 - (\mathrm{x_{B2}} - \mathrm{x_{B1}})^2}\,, \tag{3.3}$$

přičemž velikost posunutí, tedy vzdálenost vz_0, je rovno absolutní hodnotě rozdílu souřadnic y původního a posunutého bodu $y_{B2'}$ a y_{B2}.



Obrázek 3.12: Znázornění situace, kdy je třeba provést posunutí bodu B2 tak, aby vzdálenost bodů průjezdu byla rovna dvojnásobku poloměru otáčení

Postup výpočtu délky celé otočky je rozdělen na tři situace, kdy polovina horizontální vzdálenosti v je menší, rovna či větší než poloměr otáčení.

1) Vzdálenost mezi vstupními body na ose x je menší než poloměr otáčení

   Trasa otočky skládá ze tří částí tvořených oblouky, kdy levý a pravý oblouk otočky mají střed v bodech $S1$, $S2$ a střední oblouk v bodu S0, viz obrázek 3.13. V rámci algoritmu jsou vypočítány souřadnice bodů $S1$, kdy x_{S1} je rovno rozdílu x_{B1} a r_{ot}, y_{S1} je rovno y_{B1}, a $S2$, jehož souřadnice x_{S2} je rovna součtu x_{B2} a r_{ot}, y_{S2} je rovno y_{B2}. Střed úsečky $|S1S2|$, bod M se souřadnicemi $\left[\frac{x_{S1}+x_{S2}}{2}; \frac{y_{S1}+y_{S2}}{2}\right]$, je vzor bodu $S0$, který vznikne posunutím ve směru vektoru $\overrightarrow{n_u}$ o vzdálenost rovnou délce úsečky $|S0M|$, jenž lze vyjádřit jako

$$|S0M| = \sqrt{(2r)^2 - \left(\frac{|S1M|}{2}\right)^2}, \tag{3.4}$$

kde

$$|S1M| = \sqrt{(x_M - x_{S1})^2 + (y_M - y_{S1})^2}. \tag{3.5}$$

Vektor $\overrightarrow{n_u} = (-u_2; u_1)$ je normálovým vektorem ke směrovému vektoru $\vec{u}$ přímky procházející body $S1$ a $S2$, velikost a směr translace M do $S0$ je dán součinem prvků $\overrightarrow{n_u}$ s podílem délky úsečky $|S0M|$ a jeho velikosti, tedy

$$S0 = M + \overrightarrow{n_u}\frac{|S0M|}{|\overrightarrow{n_u}|}. \tag{3.6}$$

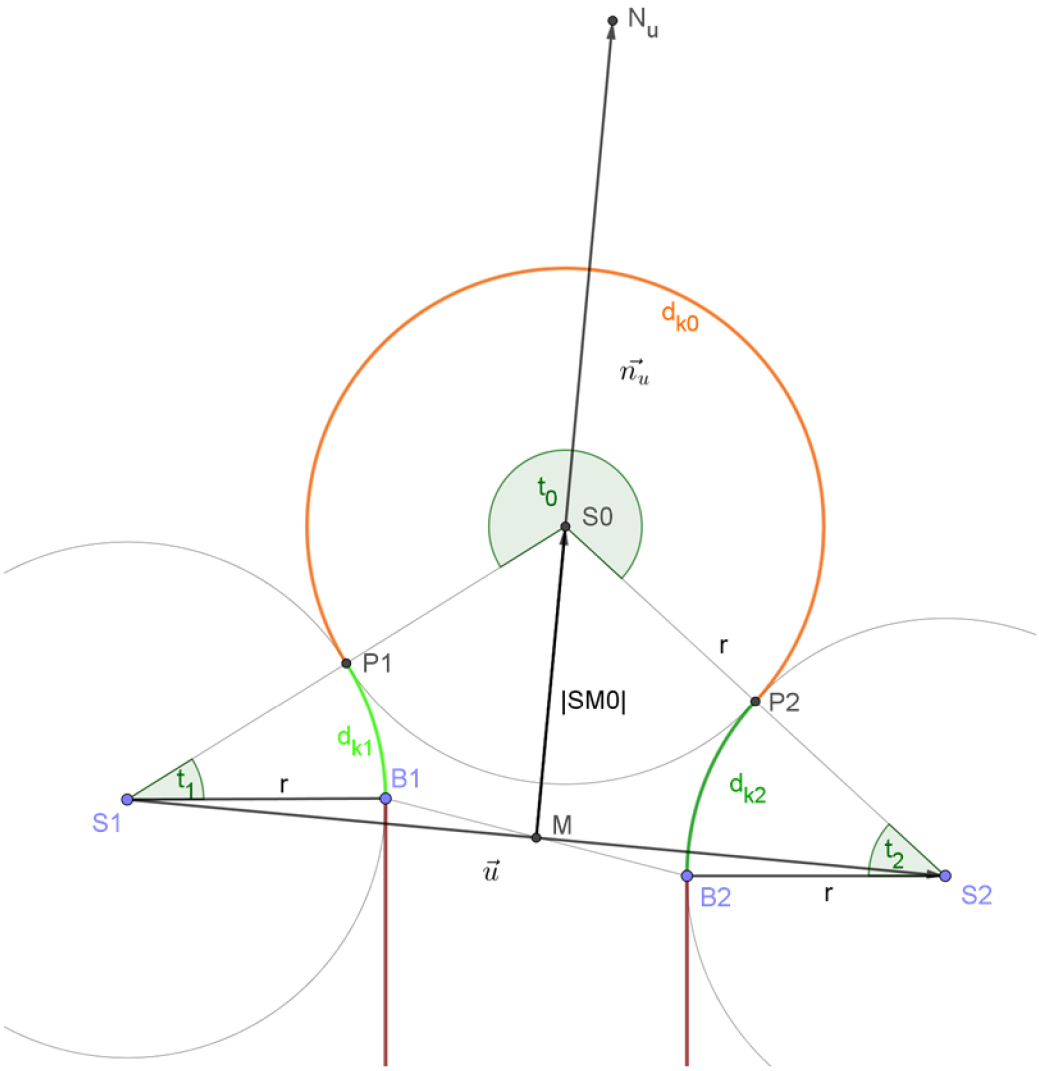
V dalším kroku jsou vypočítány souřadnice inflexních bodů $P1$ a $P2$, ve kterých dochází k přechodu mezi oblouky kruhových výsečí tvořícími části pojezdové trasy, a velikosti příslušných středových úhlů t_0, t_1 a t_2 v radiánech, jenž odpovídají odchylkám vektorů ramen těchto úhlů vyjádřených jako

$$\cos\varphi = \frac{\vec{u}\vec{v}}{|\vec{u}||\vec{v}|}, \varphi\epsilon\langle 0°; 180°\rangle, \tag{3.7}$$

přičemž pro úhel t_1 lze místo vektorů dosadit $\vec{u} = B1 - S1$ a $\vec{v} = P1 - S1$, a tak

$$t_1 = \operatorname{acos}\left(\frac{\left((x_{B1} - x_{S1})(x_{P1} - x_{S1}) + (y_{B1} - y_{S1})(y_{P1} - y_{S1})\right)}{\sqrt{(x_{B1} - x_{S1})^2 + (y_{B1} - y_{S1})^2}\sqrt{(x_{P1} - x_{S1})^2 + (y_{P1} - y_{S1})^2}}\right). \tag{3.8}$$

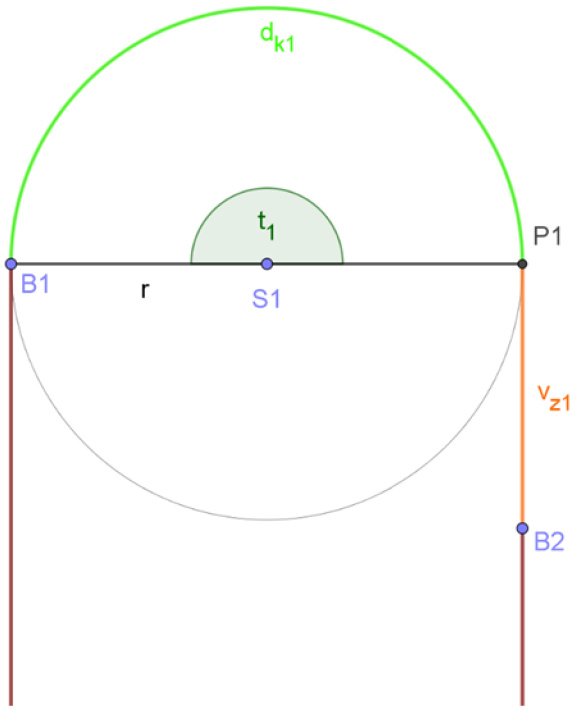
Analogicky je dopočítána velikost úhlu t_2 i t_0, avšak u něj je třeba zjištěnou velikost odečíst od 2π, protože se jedná o konkávní úhel tvořící doplněk spočítaného konvexního úhlu $\measuredangle S2S0S1$, což vyplývá z vlastností vzorce 3.7. Celková délka otočky činí suma délek jednotlivých obloukových částí trasy, které jsou vyjádřeny součinem poloměru otáčení a velikostí příslušného úhlu, a vzdálenosti vz_0 nabývající hodnoty 0 při zachování vertikální polohy obou bodů, či nenulové hodnoty v případě, že $|B1B2| > 2r$.



Obrázek 3.13: Znázorněni otočky při vzdálenosti mezi vstupními body na ose x menší, než je poloměr otáčení

2) Vzdálenost mezi vstupními body na ose x je rovna poloměru otáčení

Trasa otočky se skládá pouze z jednoho oblouku kruhové výseče s úhlem o velikosti π rad, pokud je $|x_{B2} - x_{B1}| = r_{ot}$. Oblouk je vymezen body $B1$, $B2$ pokud $y_{B1} = y_{B2}$, v případě, že $y_{B1} \neq y_{B2}$, bude mezní body tvořit bod průjezdu s vyšší absolutní hodnotou souřadnice y a bod $P1$. Například když $|y_{B1}| > |y_{B2}|$, jak je znázorněno na obrázku 3.14, bude mít bod $P1$ souřadnice$[x_{B2}; y_{B1}]$. Dále je třeba zjistit vzdálenost vz_1 odpovídající délce úsečky $|B2P1|$, tedy $vz_1 = |y_{B1} - y_{B2}|$, a tu přičíst k délce oblouku $r_{ot}\pi$, čímž je spočítána celková délka trasy otočky.



Obrázek 3.14: Znázorněni otočky při vzdálenosti mezi vstupními body na ose x rovné poloměru otáčení

3) Vzdálenost mezi vstupními body na ose x je větší než poloměr otáčení

Tato varianta operuje se skladbou otočky ze dvou oblouků a rovnou spojnicí mezi nimi, přičemž oba oblouky jsou směřovány do prostoru mezi vybranými dílčími trasami pojezdu, jejich středy budou tedy vytvořeny opačně než v první situaci, viz obrázek 3.15. Souřadnice středů $S1$, $S2$ nabývají hodnot $[x_{B1} + r_{ot}; y_{B1}]$ a $[x_{B2} - r_{ot}; y_{B2}]$, součet úhlů t_1, t_2 je roven π. Pokud $|B1S1| \parallel |B2S2|$, tak platí, že $|\sphericalangle B2S2S1| = |\sphericalangle B1S1S2|$, a vzhledem k tomu, že obě kružnice, na nichž leží oblouky otočky, mají stejný poloměr, tak společná tečna může nabývat dvou poloh stejného typu, přičemž u jedné z nich je tečna rovnoběžná se spojnicí jejich středů, úsečkou $|S1S2|$ a zároveň spojnice tečných bodů $P1$, $P2$ s odpovídajícími středy, tedy $|S1P1|$ a $|S2P2|$, jsou kolmé k $|S1S2|$, tudíž

$$|\sphericalangle B1S1S2| = 2\pi - \frac{\pi}{2} - t_1 \ \wedge |\sphericalangle B1S1S2| = \frac{\pi}{2} + t_2, \tag{3.9}$$

po úpravě

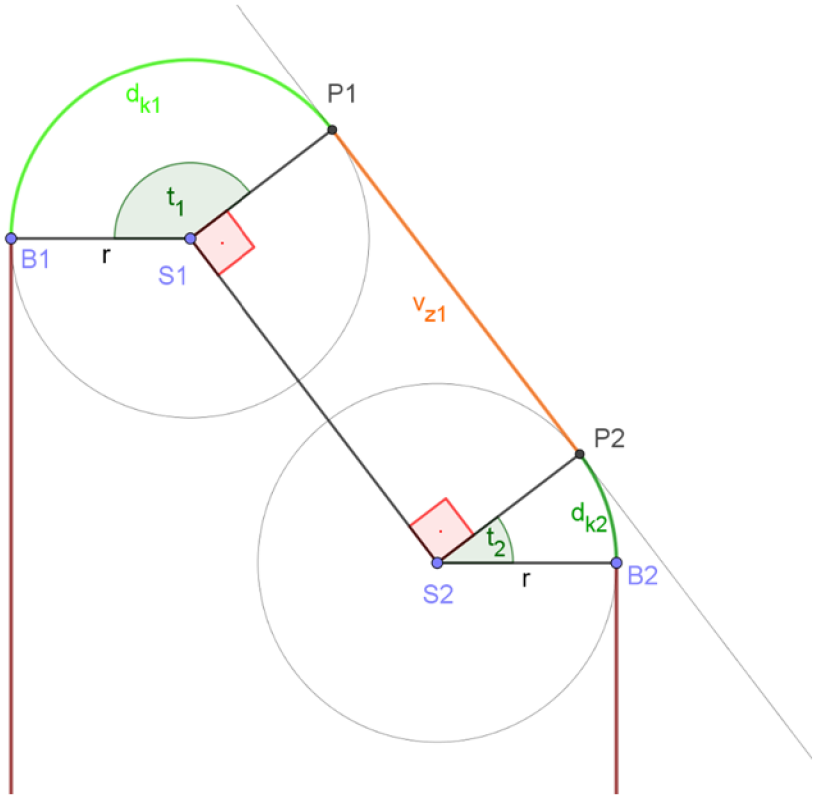
$$2\pi - \frac{\pi}{2} + t_1 = \frac{\pi}{2} + t_2 \Longrightarrow t_1 + t_2 = 2\pi - \frac{\pi}{2} - \frac{\pi}{2}, \tag{3.10}$$

tedy

$$t_2 + t_2 = \pi. \tag{3.11}$$

Celková délka trasy otočky je rovna součtu $r_{ot}\pi$ a vz_1, kde

$$vz_1 = \sqrt{(x_{S1} - x_{S2})^2 + (y_{S1} - y_{S2})^2}. \tag{3.12}$$



Obrázek 3.15: Znázorněni otočky při vzdálenosti mezi vstupními body na ose x větší, než je poloměr otáčení

Ukázka zdrojového kódu 3.9: Funkce pro výpočet délky otočky

```
1  function [vzdalenost]=f_So_ot_vz1(xB,yB,r_ot,sm)
2
3  %seřazení dvou bodů průjezdu tak, aby jako první bod byl brán bod vlevo (s
   menší hodnotou x-ové souřadnice)
   [xB1,px]=min([xB(1),xB(2)]);
4  xB2=xB(3-px);
5  yB1=yB(px);
6  yB2=yB(3-px);
7  %výpočet horizontální vzdálenosti mezi body průjezdu (vzdálenost na ose x)
   v=abs(xB(2)-xB(1));
8
9  if sqrt((xB(2)-xB(1))^2+(yB(2)-yB(1))^2)>2*r_ot &
   v/2<r_ot
```

```
10          %pokud je otočka nad body průjezdu
            if sm==1
11              %pokud je souřadnice y bodu B1 větší než bodu B2
                if yB(1)>yB(2)
12                  %uložení původní y-ové souřadnice bodu B2
                    yBold=yB(2);
13                  %výpočet nové hodnoty souřadnice y bodu B2
                    yB(2)=yB(1)-sqrt(4*r_ot^2-v^2);
14                  %výpočet vzdálenosti, o kterou je bod B2 posunut ve směru
                    osy y
                    vz0=abs(yB(2) - yBold);
15              elseif yB(2)>yB(1)
16                  yBold=yB(1);
17                  yB(1)=yB(2)-sqrt(4*r_ot^2-v^2);
18                  vz0=abs(yB(1) - yBold);
19              else
20                  vz0=0
21              End
22          %pokud je otočka pod body průjezdu
            elseif sm==0
23              if yB(2)<yB(1)
24                  yBold=yB(1);
25                  yB(1)=yB(2)+sqrt(4*r_ot^2-(xB(2)-xB(1))^2);
26                  vz0=abs(yB(1) - yBold);
27              elseif yB(1)<yB(2)
28                  yBold=yB(2);
29                  yB(2)=yB(1)+sqrt(4*r_ot^2-(xB(2)-xB(1))^2);
30                  vz0=abs(yB(2) - yBold);
31              else
32                  vz0=0;
33              end
34          end
35      else
36          vz0=0;
37      end
38
```

```
39  if v/2<r_ot
40      %střed kružnice levé části otočky (je posunut o hodnotu poloměru otá-
        čení doleva)
        S1=[xB1-r_ot,yB1];
41      %střed kružnice pravé části otočky (je posunut o hodnotu poloměru
        otáčení doprava)
        S2=[xB2+r_ot,yB2];
42      %střed spojnice úsečky |S1S2|
        M=(S1+S2)/2;
43      %délka úsečky |S0M|, tedy vzdálenost hledaného středu kružnice S0
        (kružnice střední části otočky) a bodu M
        S0M=sqrt(4*r_ot^2-(M(1)-S1(1))^2-(M(2)-S1(2))^2);
44      S0M=real(S0M);
45      %vektor úsečky |S1S2|
        u=S2-S1;
46      %normálový vektor k vektoru u
        nu=[-u(2),u(1)];
47      %výpočet souřadnic středu S0 posunutím bodu M ve směru normálového
        vektoru, přičemž velikost posunutí je rovna podílu vzdálenosti bodu M
        od bodu S0, což je hodnota S0M, a velikosti tohoto vektoru
        S0=M+nu/sqrt(nu(1)^2+nu(2)^2)*S0M;
48      %výpočet souřadnic bodu, kde trasa přechází z oblouku kružnice se
        středem v S1 do oblouku kružnice se středem v S0
        P01=(S0+S1)/2;
49      %výpočet souřadnic bodu, kde trasa přechází z oblouku kružnice se
        středem v S0 do oblouku kružnice se středem v S2
        P02=(S0+S2)/2;
50      %výpočet velikosti úhlu, který svírají úsečky |S0P01| a |S0P02| (úhel
        výseče kružnice se středem S0, která je střední částí dráhy otočky)
        t0=2*pi-acos(((P02(1)-S0(1))*(P01(1)-S0(1))+(P02(2)-
            S0(2))*(P01(2)-S0(2)))/(sqrt((P02(1)-
            S0(1))^2+(P02(2)-S0(2))^2)*sqrt((P01(1)-
            S0(1))^2+(P01(2)-S0(2))^2)));
51      %výpočet délky výseče střední části trasy otočky
        dk0=r_ot*(t0);
52      %výpočet velikosti úhlu, který svírají úsečky |S1B1| a |S1P01| (úhel
        výseče kružnice se středem S1, která je levou částí dráhy otočky)
        t1=acos(((xB1-S1(1))*(P01(1)-S1(1))+(yB1-
            S1(2))*(P01(2)-S1(2)))/(sqrt((xB1-S1(1))^2+(yB1-
            S1(2))^2)*sqrt((P01(1)-S1(1))^2+(P01(2)-
            S1(2))^2)));
53      dk1=r_ot*(t1);
54      %výpočet velikosti úhlu, který svírají úsečky |S2B2| a |S2P02| (úhel
        výseče kružnice se středem S2, která je pravou částí dráhy otočky)
```

```
        t2=acos(((xB2-S2(1))*(P02(1)-S2(1))+(yB2-
           S2(2))*(P02(2)-S2(2)))/(sqrt((xB2-S2(1))^2+(yB2-
           S2(2))^2)*sqrt((P02(1)-S2(1))^2+(P02(2)-
           S2(2))^2)));
55      dk2=r_ot*(t2);
56      %výpočet celkové délky otočky
        vzdalenost=dk0+dk1+dk2+vz0;
57  elseif v/2==r_ot
58      %pokud je polovina vzdálenosti bodů na ose x rovna poloměru
        otáčení, tak dráhu otočení tvoří půlkružnice, jejíž počáteční
        i koncový bod mají stejnou polohu na ose y, která je rovna
        poloze bodu s vyšší absolutní hodnotou y-ové souřadnice, je
        tedy třeba dopočítat vzdálenost k poloze druhého bodu průjezdu
        na ose y
        vz1=abs(yB2-yB1);
59      vzdalenost=pi*r_ot+vz1;
60  else
61      %střed kružnice levé části otočky (je posunut o hodnotu poloměru otá-
        čení doprava)
        S1=[xB1+r_ot,yB1];
62      %střed kružnice pravé části otočky (je posunut o hodnotu poloměru
        otáčení doleva)
        S2=[xB2-r_ot,yB2];
63      %velikost velikosti úsečky |S1S2|, potažmo |P1P2|
        vz1=sqrt((S1(1)-S2(1))^2+(S1(2)-S2(2))^2);
64      vzdalenost=pi*r_ot+vz1;
65  end
```

3.7 Funkce selekce

Implementované selekční funkce mají podobné, v některých případech i stejné, části zdrojového kódu zaměřeného na výběr jedinců k dalšímu zpracování, a tak u první popisované funkce zaměřené na pořadovou selekci bude uveden kompletní zdrojový kód a funkce bude popsána podrobněji, u dalších již jen rozdílné části. Vstupní parametry jsou pro všechny funkce stejné, jedná se o soubor chromozomů *populace* o velikosti N_p a list zahrnující jejich ohodnocení fit. Výstupem je skupina vybraných chromozomů *P_vyber* s polovičním počtem prvků oproti vstupnímu souboru. V rámci funkce je nejprve získán počet vybíraných chromozomů, jež je roven $\frac{N_p}{2}$, poté je sestupně seřazen list fit obsahující hodnoty získané hodnotící funkcí, přičemž funkce *sort* provede řazení hodnot se zachováním informace o původním pořadí každého

prvku, které jsou zaznamenány do listu por. Poté je vygenerován list hodnot $p\_rank = \{N\_p, N\_p - 1, N\_p - 2, \dots, 1\}$ v posloupnosti klesající o 1 v rozmezí hodnoty počtu členů N_p až 1, ze které je vydělením každého členu listu sumou hodnot všech členů, dle vzorce 3.13, získán list pravděpodobností výběru úměrných hodnotě nového pořadí seřazených chromozomů p_v, na základě jehož členů je sestavena stupnice r tvořící nedílnou součást samotného výběru chromozomů, tj.

$$p_{v_i} = \frac{p_{rank_i}}{\sum_{j=0}^{N_p} p_{rank_j}}, i = \{1, \dots, N_p\}. \tag{3.13}$$

Sestavení stupnice pro výběr probíhá pomocí *for* cyklu, kdy proměnná i nabývá hodnot od 2 do počtu prvků listu p_v, přičemž hodnota prvku stupnice je rovna součtu i posledních prvků p_v, viz vzorec 3.14, po skončení cyklu je na konec škály přidána 0:

$$r_{j-1} = \sum_{i=j}^{N_p} p_{v_i}, j = \{2, \dots, N_p\}. \tag{3.14}$$

Například u listu ohodnocení $fit = [9, 15, 7, 11]$, u kterého je níže pro lepší názornost je k hodnotám v řádku 1 zapsáno i pořadí jejich polohy v listu do řádku 2, po seřazení hodnot sestupně dochází ke změnám pozic hodnot původního pořadí

$$fit = \begin{bmatrix} 9 & 15 & 7 & 11 & hodnoty \\ 1 & 2 & 3 & 4 & pořadí \end{bmatrix} \rightarrow$$

$$\rightarrow fit_{sort} \begin{bmatrix} 15 & 11 & 9 & 7 & hodnoty_{sort} \\ 2 & 4 & 1 & 3 & por \end{bmatrix},$$

tedy $por = [2, 4, 1, 3]$. Protože $N_p = 4$, $p_{rank} = [4,3,2,1]$, poté jsou získány hodnoty p_v, kdy

$$p_v = \left[\frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}\right],$$

z nichž jsou dosazením do vzorce 14 vypočítány hodnoty stupnice r, která je nakonec doplněna o hodnotu 0:

$$r = [0{,}6; 0{,}3; 0{,}1] \rightarrow [0{,}6; 0{,}3; 0{,}1; 0].$$

Následně jsou v cyklu for probíhajícího n-krát, kdy n je rovno polovině počtu vstupních řešení, vybírány chromozomy pomocí náhodně vygenerované hodnoty a z rozmezí $\langle 0; 1 \rangle$, přičemž je tato hodnota porovnávána se členy stupnice a dochází k hledání nejmenší hodnoty stupnice, která je větší nebo rovna hodnotě a, avšak zaznamenána není nalezená hodnota ale její poloha (pořadí) r_m na stupnici. Finální výběr probíhá nalezením hodnoty v listu por s indexem r, která poslouží jako index vybíraného řešení v rámci vstupní populace.

Pro názornost bude uvedeno pokračování příkladu, kdy je vygenerována hodnota $a = 0{,}35$, při které podmínka $r \geq a$ platí pro prvky $r_1 = 0{,}6$ a $r_2 = 0{,}3$, $\min(r_1, r_2) = r_2$, což je člen s druhým pořadím, tedy $r_m = 2$, tato hodnota je dosazena jako index $populace(por(r_m))$, po dosazení $populace(por(2))$, kdy $por(2) = 4$, poté $populace(4)$, tedy vybraný chromozom se ve vstupním souboru řešení nachází na čtvrtém místě.

```
1   function [P_vyber]=f_GAp03_sel_rank1(populace,fit)
2
3    %počet řešení pro výběr selekcí
     poc_vyber=round(size(populace,1)/2);
4    %seřazení vektoru hodnot od největší po nejmenší, získání seznamu pořadí
     nejlepších řešení ve vstupním souboru
     [~,por]=sort(fit,'descend');
5    %list klesajících hodnot po -1 od počtu členů populace do 1
     p_rank=(length(por):-1:1);
6   p_v=p_rank./sum(p_rank);
7   r=[];
8   for i=2:1:length(p_v)
9       q0=sum(p_v(i:end));
10      r=[r,q0];
11  end
12  r=[r,0];
13
14  P_vyber=[];
15  for j=1:poc_vyber
16      a=rand();
17      [~,r_m]=min(r(r>=a));
18      P_vyber=[P_vyber;populace(por(r1),:)];
19  end
```

Selekce pořadová lineární i exponenciální se liší od standardní pořadové pouze ve způsobu vytvoření listu p_v, přičemž jeho prvky vytvořeny dle vzorce 3.15 a 3.16, u lineární pořadové selekce je zaveden parametr SP představující selektivní tlak nabývající hodnot z intervalu $\langle 1; 2\rangle$ a u exponenciální se zde vyskytuje parametr c ovlivňující rozložení pravděpodobnosti výběru jednotlivých řešení, pro který platí $0 < c < 1$, tedy

$$p1_i = 2 - SP + \frac{2(SP-1)\left(P_{rank_i} - 1\right)}{|P_{rank}| - 1}, \tag{3.15}$$

$$p1_i = \frac{c^{|P_{rank}| - P_{rank_i}}}{\sum_{j=0}^{|P_{rank}|} c^{|P_{rank}| - P_{rank_j}}}. \tag{3.16}$$

Ruletová selekce je založena na stejném principu jako selekce pořadové, s tím rozdílem, že pravděpodobnost výběru je přímo ovlivněna hodnotou fit nikoliv jejím pořadím vůči celku, hodnoty prvků listu p_v jsou rovny podílu odpovídajících hodnot fit a součtu všech hodnot fit, viz vzorec 3.17

$$p1_i = \frac{fit_i}{\sum_{j=0}^{N} fit_j}. \tag{3.17}$$

V algoritmu je také implementován elitářský výběr ve dvou variantách, při kterém probíhá výběr stanovené části chromozomů s nejvyšším ohodnocením, tedy n nejlepších řešení, následně je počet zbývající do poloviny velikosti celé populace $\frac{N_p}{2} - n$ vybrán pomocí pořadové či ruletové selekce.

Stochastická univerzální metoda pracuje se stejnou pravděpodobností výběru jako ruletová selekce, rozdíl je však ve způsobu výběru, kdy je nejprve vygenerována náhodná hodnota u z intervalu $\langle 0; \frac{\sum_{j=0}^{N} fit_j}{N_{pop}/2} \rangle$, přičemž hodnoty v, na jejichž základě probíhá výběr, jsou vypočítány dle vzorce 3.18

$$v_i = \left(\frac{N_{pop}}{2} - i\right) * \frac{\sum_{j=0}^{N} fit_j}{\frac{N_{pop}}{2}} + u, i = \left\{1, \dots, \frac{N_{pop}}{2}\right\}. \tag{3.18}$$

Ukázka zdrojového kódu 3.10: Stochastická univerzální metoda – ukázka části týkající se výběru chromozomů ze vstupní skupiny

```
1    function [P_vyber]=f_GAp03_sel_sus(populace,fit)
⋮    ⋮
12   u=rand()*sum(p_v)/poc_vyber;
13    %list obsahující vygenerované hodnoty pro výběr chromozomů
     v=[poc_vyber-1:-1:0].*sum(p_v)/poc_vyber+u;
14   P_vyber=[];
15   for i=1:length(v)
16       [~,r1]=min(r(r>=v(i)));
17       P_vyber=[P_vyber;populace(r1,:)];
18   end
```

Turnajová selekce je poslední metodou zahrnutou do algoritmu, od ostatních se liší ve způsobu výběrů, kdy jsou dva či více prvků, v závislosti na zvolené variantě, ze vstupní skupiny řešení náhodně vybrány a porovnány na základě jejich ohodnocení a následně je zvolen vítězný prvek, tedy ten s nejvyšší hodnotou. Vybraný chromozom může být ze vstupní skupiny odstraněn, aby nedocházelo k jeho opakovaným výběrům, nebo v ní ponechán.

Ukázka zdrojového kódu 3.11: Turnajová selekční metoda s porovnáním dvou prvků

```
1   Function [P_vyber]=f_GAp03_sel_tourna-
    ment2(P_poc,fit,o,P_npop)
2
3   poc_vyber=round(size(populace,1)/2);
4   P_vyber=[];
5   for i=1:poc_vyber
6         %počet chromozomů N_p ve vstupním souboru
          q=size(populace,1);
7         %vygenerování dvou náhodných čísel z intervalu <0, N_p>
          t=randperm(q,2);
8         %pořadí vybraných prvků z listu fit s největší hodnotou v rámci výběru
          [~,vs0]=max(fit([t]));
9         %zjištění hodnoty čísla ze vygenerované skupiny t, na základě pořadí
          získaného v předchozím kroku, tato hodnota poslouží dále jako index
          pro výběr chromozomu ze vstupní skupiny
          vs=t(vs0);
10        P_vyber(i,:)=[populace(vs,:)];
11        populace(vs,:)=[];
12        fit(vs)=[];
13  end
```

3.8 Křížení

Pro implementaci do algoritmu byly vybrány dva typy křížení, se zachováním pořadí a na základě pozice, jejichž aplikací nedochází ke ztrátě prvků v rámci jednotlivých řešení, tedy genů chromozomů, ani k jejich duplikaci jako u bodového či fúzního křížení. Obě funkce mají dva vstupy, populaci získanou selekcí P_{vyber} a pravděpodobnosti křížení P_c, výstupem je generace potomků. Pro křížení jsou vybrány dva chromozomy ze vstupní populace, tzv. rodičové, ze kterých jsou s pravděpodobností P_c vytvořeni potomci. V implementovaném algoritmu probíhá výběr tak, že jsou v rámci for cyklu vždy zvoleny dvojice po sobě jdoucích prvků $P_{vyber_{2k-1}}$ a $P_{vyber_{2k}}$, kdy

$k = \{1, .., N_p/2\}$, před tím je nejprve vygenerován list o sto prvcích s hodnotami 1 nebo 0, přičemž poměr nenulových a nulových prvků odpovídá P_c. Poté, již v rámci cyklu, je vygenerováno náhodné číslo b z intervalu $\langle 1; 100 \rangle$ sloužící pro výběr prvku z listu a. Má-li prvek s indexem b hodnotu 1, dochází ke křížení, v opačném případě jsou do populace potomků zařazeni vybraní rodiče.

Při křížení se zachováním pořadí dochází k výběru genů rodičů vygenerováním dvou náhodných bodů křížení kr_1 *a* kr_2 v podobě čísel z intervalu $\langle 1, n \rangle$, kde n je počet genů v chromozomu, seřazených vzestupně dle velikosti, tyto body určují rozsah pořadí genů, které budou přeneseny z rodiče 1 na potomka 1, přičemž jejich pozice v novém chromozomu je stejná jako v původním. Po zápisu genů z rodiče 1, řetězec rod_{kr}, jsou geny rodiče 2 ($rod2$) se stejnou hodnotou vymazány, tedy je jim přiřazena hodnota 0. V dalším kroku dochází k vytvoření řetězců $rod2_1$ a $rod2_3$ obsahujících nenulové prvky chromozomu $rod2$ s indexem pozice menším než kr_2 a větším než kr_2.Pokud je počet volných míst v chromozomu potomka 1 s indexem větším než kr_2 větší, než délka řetězce $rod2_3$ N_{r23}, tedy $n - kr_2 > N_{r23}$, tak je tato část doplněna všemi prvky z $rod2_3$, dopočítán zbylý počet volných míst $r_{rp} = n - kr_2 - N_{r23}$a doplněn prvky s indexem od 1 do r_{rp} včetně z řetězce $rod2_1$. Zbylá volná místa v chromozomu potomek 1 jsou od jeho počátku doplněna prvky $rod2_1$ s indexem větším než r_{rp}. V případě, že je počet volných míst v chromozomu potomka 1 s indexem větším než kr_2 roven délce řetězce $rod2_3$, jsou na tyto pozice dosazeny všechny prvky $rod2_3$ a volná místa s indexem menším doplní prvky $rod2_1$. Tvorba druhého potomka je založena na stejném principu, pouze role rodičů se obrací.

Ukázka zdrojového kódu 3.12: Křížení se zachováním pořadí

```
1  function [potomcik]=f_GAp04cross_order(P_vyber,P_c)
2
3    %počet prvků v každém řešení ze vstupní skupiny (genů v chromozomu)
   n=size(P_vyber,2);
4  potomcik=[];
5    %vytvoření listu o délce 100 prvků obsahující pouze hodnoty 1 a 0 v poměru
     odpovídajícímu pravděpodobnosti křížení, například pokud P_c=0,65, tak 65
     prvků má hodnotu 1 a zbylých 35 hodnotu 0
   a=[ones(1,round(P_c*100)),zeros(1,round((1-P_c)*100))];
6
7  for k1=1:floor(size(P_vyber,1)/2)
```

```
8          %vygenerování náhodného čísla z intervalu <1;100>
       b=randperm(100,1);
9          %pokud prvek z listu a o pořadí aa roven 1, tak dochází ke křížení
       if a(b)==1
10             %rodič 1, lichý chromozom o pořadí 2*k1-1, kde k1=N_p/2
           rod1=(P_vyber(2*k1-1,:));
11             %rodič 2, sudý chromozom o pořadí 2*k1, kde k1=N_p/2
           rod2=(P_vyber(2*k1,:));
12             %výběr dvou náhodných čísel (bodů křížení) kr1 a kr2 z intervalu
               <1,n>, kde n je délka chromozomu (počet prvků), seřazených dle
               velikosti vzestupně
           kr=sort(randperm(n,2));
13             %výběr prvků rodiče 1 s indexem v rozsahu od kr1 do kr2
           rod1_kr=rod1(kr(1):kr(2));
14             %vytvoření potomka 1 v podobě nulového matice o velikost 1×n,
               nula představuje volné místo k zápisu genů (prvků) rodičů
           pot1=zeros(1,n);
15             %dosazení vybraných prvků z rodiče 1 do potomka 1, přičemž indexy
               prvků, tedy jejich pozice, zůstávají zachovány
           pot1(kr(1):kr(2))=rod1_kr;
16             %nahrazení prvků rodiče 2 s hodnotou nacházející se ve výběru z
               rodiče 1 hodnotou 0
           for k2=1:length(rod1_kr)
17                 najdi=find(rod2==rod1_kr(k2));
18                 rod2(najdi)=0;
19             end
20
21             rod2_3=rod2(kr(2)+1:n);
22             %výběr nenulových prvků z rodiče 2 s pořadím větším, než je druhý
               bod křížení
           rod2_3=rod2_3(rod2_3~=0);
23             rod2_1=rod2(1:kr(2));
24             %výběr nenulových prvků z rodiče 2 s pořadím menším, než je druhý
               bod křížení
           rod2_1=rod2_1(rod2_1~=0);
25
26             %pokud je počet volných míst v chromozomu potomka v části za
               druhým bodem křížení (n-kr2) větší než počet nenulových prvků
               rodiče 2 v odpovídající části
           if (n-kr(2))>length(rod2_3)
27                 %rozdíl mezi volnými místy v chromozomu potomka v části za
                   druhým bodem křížení a počtu nenulových prvků rodiče 2 za
                   druhým bodem křížení
```

```
                r_rp=(n-kr(2))-length(rod2_3);
28              %dosazení nenulových prvků z části za kr2 u rodiče 2 do stejné části potomka 1, přičemž jsou prvky řazeny za sebe bez zachování pozice
                pot1(kr(2)+1:kr(2)+length(rod2_3))=rod2_3;
29              %doplnění zbývajících volných míst v části za kr2 u potomka1 nenulovými geny z části před kr2 u rodiče 2, vybráno je prvních r_rp genů
                pot1((kr(2)+len-
                gth(rod2_3)+1):n)=rod2_1(1:r_rp);
30              %nahrazení prvních r_rp genů z rodiče 2, které již byly do potomka 1 dosazeny, hodnotou nula
                rod2_1(1:r_rp)=0;
31              %odstranění nulových prvků
                rod2_1=rod2_1(rod2_1~=0);
32              doplnění zbývajících genu z části před kr2 u rodiče do volných míst potomka 1
                pot1(1:kr(1)-1)=rod2_1;
33          end
34          %pokud je počet volných míst v chromozomu potomka v části za druhým bodem křížení (n-kr2) roven počtu nenulových prvků rodiče 2 v odpovídající části
            if (n-kr(2))==length(rod2_3)
35              %dosazení nenulových prvků z části za kr2 u rodiče 2 do stejné části potomka 1
                pot1(kr(2)+1:n)=rod2_3;
36              %dosazení nenulových prvků z části před kr2 u rodiče 2 do stejné části potomka 1
                pot1(1:kr(1)-1)=rod2_1;
37          end
38          %potomek 2 - proces tvorby probíhá stejným způsobem jako u potomka 1, pouze role rodičů jsou prohozené
39          rod1=(P_vyber(2*k1-1,:));
40          rod2=(P_vyber(2*k1,:));
41          rod2_kr=rod2(kr(1):kr(2));
42          pot2=zeros(1,n);
43          pot2(kr(1):kr(2))=rod2_kr;
44          for k2=1:length(rod2_kr)
45              najdi=find(rod1==rod2_kr(k2));
46              rod1(najdi)=0;
47          end
48
```

```
49          rod1_3=rod1(kr(2)+1:n);
50          rod1_3=rod1_3(rod1_3~=0);
51          rod1_1=rod1(1:kr(2));
52          rod1_1=rod1_1(rod1_1~=0);
53
54          if (n-kr(2))>length(rod1_3)
55              aa=(n-kr(2))-length(rod1_3);
56              pot2(kr(2)+1:kr(2)+length(rod1_3))=rod1_3;
57              pot2((kr(2)+len-
                gth(rod1_3)+1):n)=rod1_1(1:aa);
58              rod1_1(1:aa)=0;
59              rod1_1=rod1_1(rod1_1~=0);
60              pot2(1:kr(1)-1)=rod1_1;
61          end
62          if (n-kr(2))==length(rod1_3)
63              pot2(kr(2)+1:n)=rod1_3;
64              pot2(1:kr(1)-1)=rod1_1;
65          end
66       %pokud nedochází ke křížení, do populace potomků jsou zařazení rodiče
        else
67          pot1=(P_vyber(2*k1-1,:));
68          pot2=(P_vyber(2*k1,:));
69      end
70      potomcik=[potomcik;pot1;pot2];
71  end
```

Křížení na základě pozice se od předchozího typu liší způsobem výběru genů, který je realizován na základě listu r_{num} náhodně vygenerovaných čísel z rozsahu $\langle 1; n\rangle$ obsahujícího náhodný počet prvků v rozmezí $\langle 1; n-1\rangle$. Po vytvoření r_{num} jsou chromozomu potomek 1 přiřazeny geny rodiče 1 nacházející se na pozicích s indexem dle hodnot prvků listu r_{num}, kdy je kromě hodnoty genu přenesena, zachována, i jejich pozice. Geny rodiče 2 se stejnou hodnotou jako ty přenesené z rodiče 1 jsou vymazány, v rámci kódu je jim přiřazena hodnota 0, zbývající nenulové prvky jsou dosazeny do potomka 1. Potomek 2 vznikne stejným způsobem jen s opačným zapojením rodičů.

Ukázka zdrojového kódu 3.13: Křížení na základě pozice

```
1  function [potomcik]=f_GAp04cross_order(P_vyber,P_c)
2
3  n=size(P_vyber,2);
4  potomcik=[];
5  a=[ones(1,round(P_c*100)),zeros(1,round((1-P_c)*100))];
6
7  for k1=1:floor(size(P_vyber,1)/2)
8      b=randperm(100,1);
9      if a(b)==1
10         rod1=(P_vyber(2*k1-1,:));
11         rod2=(P_vyber(2*k1,:));
12          %vytvoření náhodného souboru čísel z rozsahu 1...počet prvků
            v chromozomu-1, soubor obsahuje náhodné množství prvků (maxi-
            málně n-1)
           r_num=randperm(n,round(rand*(n-1)));
13         pot1=zeros(1,n);
14          %výběr prvků (genů) z rodiče 1 předaných potomkovi 1 (je zacho-
            vána pozice vybraných prvků) dle náhodně vytvořeného souboru
            čísel
           pot1(r_num)=rod1(r_num);
15         for k2=1:length(rod2)
16             najdi=find(rod2==pot1(k2));
17             rod2(najdi)=0;
18         end
19          %dosazení nenulových prvků rodiče 2 na prázdná místa (prvky
            s hodnotou 0) v chromozomu potomek 1
           pot1(pot1==0)=rod2(rod2~=0);
20          %potomek 2 - proces tvorby probíhá stejným způsobem jako u po-
            tomka 1, pouze role rodičů jsou prohozené
21         rod1=(P_vyber(2*k1-1,:));
22         rod2=(P_vyber(2*k1,:));
23         pot2=zeros(1,n);
24         pot2(r_num)=rod2(r_num);
25         for k2=1:length(rod1)
26             najdi=find(rod1==pot2(k2));
27             rod1(najdi)=0;
28         end
29         pot2(pot2==0)=rod1(rod1~=0);
```

```
30        else
31            pot1=(P_vyber(2*k1-1,:));
32            pot2=(P_vyber(2*k1,:));
33        end
34        potomcik=[potomcik;pot1;pot2];
35    end
```

3.9 Mutace

Z důvodu zachování všech prvků vstupních řešení a zamezení jejich duplikaci byly vybrány k implementaci mutace výměnou a posuvná mutace, označovaná také jako mutace vložení. Vstupy v podobě pravděpodobnosti mutace P_m a populace vzniklé křížením P_{pok} i výstup populace po mutaci P_{mut} jsou stejné, taktéž první část obou algoritmů zaměřená na provedení úkonu se zadanou pravděpodobností vychází z procesu popsaného již u metod křížení.

V rámci mutace výměnou nejprve během cyklu opakujícího se dle počtu chromozomů ve vstupní populaci po splnění pravděpodobnostní podmínky dochází k vygenerování dvou náhodných čísel, k_{m1} a k_{m2} z intervalu $\langle 1; n \rangle$, kde n je délka chromozomů, která jsou využita jako indexy pro výběr genů, jenž si navzájem vymění pozici v chromozomu. Následně probíhá samotná výměna, kdy gen s indexem k_{m1} je přesunut na pozici k_{m2} a gen s indexem k_{m2} na k_{m1}.

Ukázka zdrojového kódu 3.14: Mutace výměnou

```
1   function [P_mut]=f_GAp05mut_exch(P_pok,P_m)
2
3   n=size(P_pok,2);
4   P_mut=P_pok;
5   a=[ones(1,round(P_m*100)),zeros(1,round((1-P_m)*100))];
6
7   for m=1:round(size(P_pok,1))
8       aa=randperm(100,1);
9       if a(aa)==1
10            %výběr dvou náhodných čísel (bodů mutace) k_m1 a k_m2 z inter-
              valu <1;n>, kde n je délka chromozomu (počet prvků)
              k_m=randperm(n,2);
11            A=[k_m(1),k_m(2)];
12            B=[k_m(2),k_m(1)];
```

```
13              %výměna pozice dvou genů, kdy gen s indexem k_m1 je pře-
                sunut na pozici k_m2 a gen s indexem k_m2 na k_m1
                P_mut(m,A)=P_mut(m,B);
16          end
17      end
```

Při posuvné mutaci jsou také generována dvě náhodná čísla, ale v tomto případě první číslo k_{m1} určuje index vybraného genu a k_{m2} index pozice, na kterou bude vybraný gen přesunut. Pokud je $k_{m1} > k_{m2}$, tak jsou všechny prvky chromozomu s indexem rovným nebo větším k_{m1} a menším než k_{m2} posunuty o jednu pozici dále, jejich index je zvětšen o hodnotu 1. V opačném případě dochází k posunutí prvků s větší než k_{m1} a menší nebo rovno k_{m2} o jednu pozici blíže k začátku chromozomu, tedy nastává zmenšení jejich indexu o 1. Do algoritmu je tento postup převeden tak, že po vytvoření nového chromozomu, nulové matice o velikosti $1 \times n$, je proveden zápis prvků s indexem menším než hodnota k_{m2} z původního do nového chromozomu, přičemž jejich pozice zůstává nezměněna, poté je přenesen na pozici k_{m2} gen s indexem k_{m1}, který je v původním chromozomu smazán. Nakonec jsou doplněny zbývající geny z původního chromozomu s indexem větším nebo rovno k_{m2}, který je následovně ve vstupní populaci přepsán novým chromozomem.

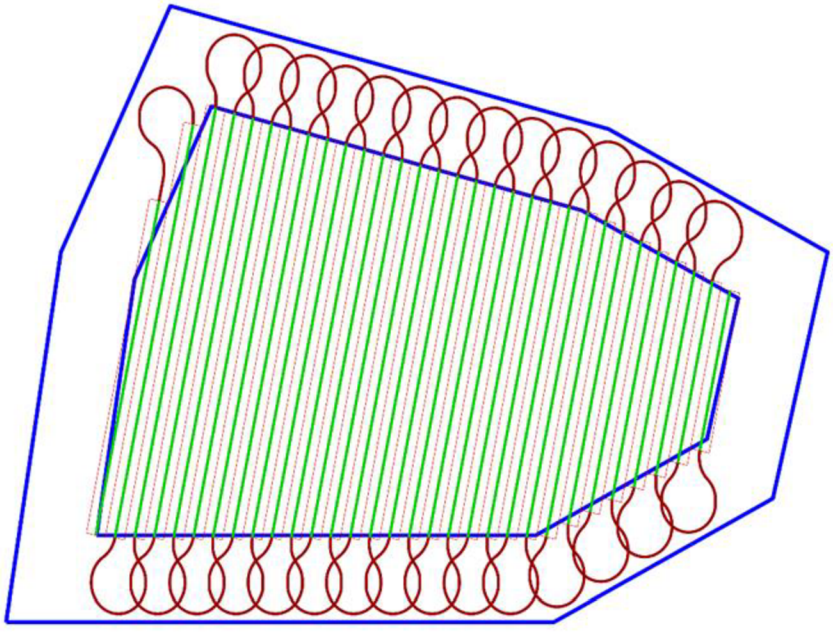
Ukázka zdrojového kódu 3.15: Posuvná mutace

```
1   function [P_mut]=f_GAp05mut_slide(P_pok,P_m)
2
3   n=size(P_pok,2);
4   P_mut=P_pok;
5   a=[ones(1,round(P_m*100)),zeros(1,round((1-P_m)*100))];
6
7   for m=1:round(size(P_pok,1))
8       aa=randperm(100,1);
9       if a(aa)==1
10              %výběr chromozomu s pořadím m ze vstupní populace
                P_mut_m=P_pok(m,:);
11              %výběr dvou náhodných čísel (původního indexu bodu mutace a no-
                vého indexu) k_m1 a k_m2 z intervalu <1;n>, kde n je délka chro-
                mozomu (počet prvků)
                k_m=randperm(n,2);
```

```
12              %pokud je původní index větší než index nové pozice
            if k_m(1)>k_m(2)
13                  %vytvoření nového chromozomu v podobě v podobě nulového
                    matice o velikost 1×n
                potomci_1=zeros(1,n);
14                  %zápis prvků s indexem menším než hodnota k_m2 z původního
                    do nového chromozomu
                potomci_1(1:k_m(2)-1)=P_mut_m(1:k_m(2)-1);
15                  %zápis prvku s indexem k_m1 z původního na pozici s indexem
                    k_m2 v novém chromozomu
                potomci_1(k_m(2))=P_mut_m(k_m(1));
16                  %odstranění přesunutého genu z původního chromozomu
                P_mut_m(k_m(1))=[];
17                  %doplnění zbývajících genů nového chromozomu prvky původ-
                    ního chromozomu (po odstranění prvku k_m1) s indexem k_m2 a
                    větším
                potomci_1((k_m(2)+1):n)=P_mut_m((k_m(2)):n-
                1);
18                  %přepsání chromozomu ve vstupní populaci novou verzí
                    zvniklou mutací
                P_mut(m,:)=potomci_1;
19          elseif k_m(1)<k_m(2)
20                  potomci_1=zeros(1,n);
21                  potomci_1(k_m(2)+1:n)= P_mut_m(k_m(2)+1:n);
22                  potomci_1(k_m(2))=P_mut_m(k_m(1));
23                  P_mut_m(k_m(1))=[];
24                  potomci_1(1:(k_m(2)-1))=P_mut_m(1:(k_m(2)-
                    1));
25                  P_mut(m,:)=potomci_1;
26          end
27      end
28  end
```

4 Vybrané výsledky a diskuse

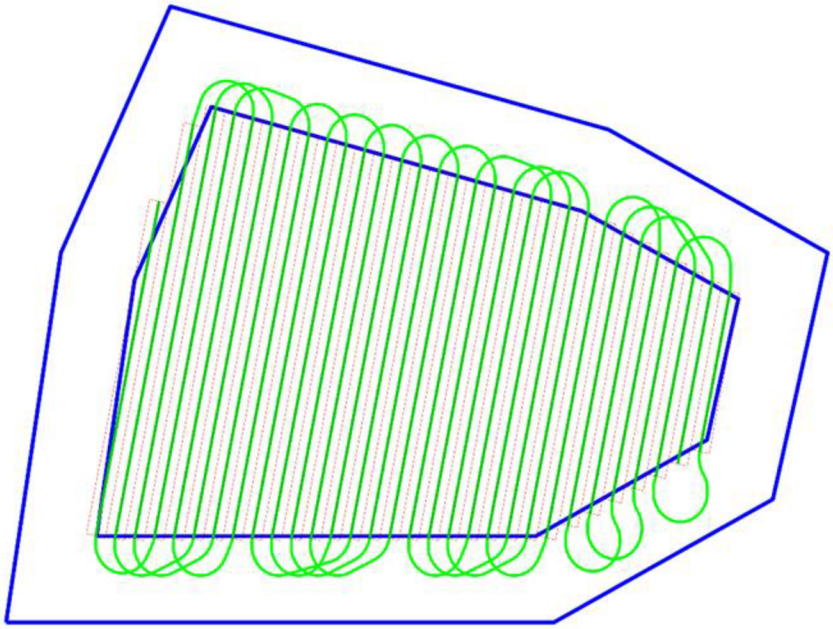
Aplikací implementovaných algoritmů byly v rámci stanovených vstupních parametrů generovány kombinace pořadí průjezdu dílčích tras pojezdu techniky po pozemku za účelem nalezení optimálního řešení, při kterém bude délka pojezdu v rámci otáčení na souvratích minimalizována. Porovnání metod zahrnutých do genetického algoritmu, tedy typů selekce, křížení a mutace, a zvolených proměnných v podobě počtu generací, velikosti populace, pravděpodobnosti mutace či křížení, bylo provedeno na vygenerovaném pozemku znázorněném včetně dílčích tras členkového pojezdu na obrázku 4.1, zvolená hodnota pro záběr pracovního nástroje byla 3,5 m a poloměr otáčení 5 m. Všechny získané hodnoty jsou průměrem padesáti opakování, tedy dále často zmiňovaná hodnota „Minimální celková délka tras při otáčení“ je průměrem minimálních hodnot získaných během jednotlivých epoch každého z padesáti průběhů celého algoritmu, zároveň je v každé generaci vypočítán průměr ze všech řešení, chromozomů, v populaci a tato hodnota je znovu průměrována vzhledem k opakovaným cyklům. Simulace byly realizovány na počítačové sestavě obsahující deseti jádrový procesor Intel Core i9 10900KF doplněný 64 GB operační paměti, uváděné údaje týkající se doby trvání jednotlivých výpočtů jsou vztaženy k této hardwarové konfiguraci.



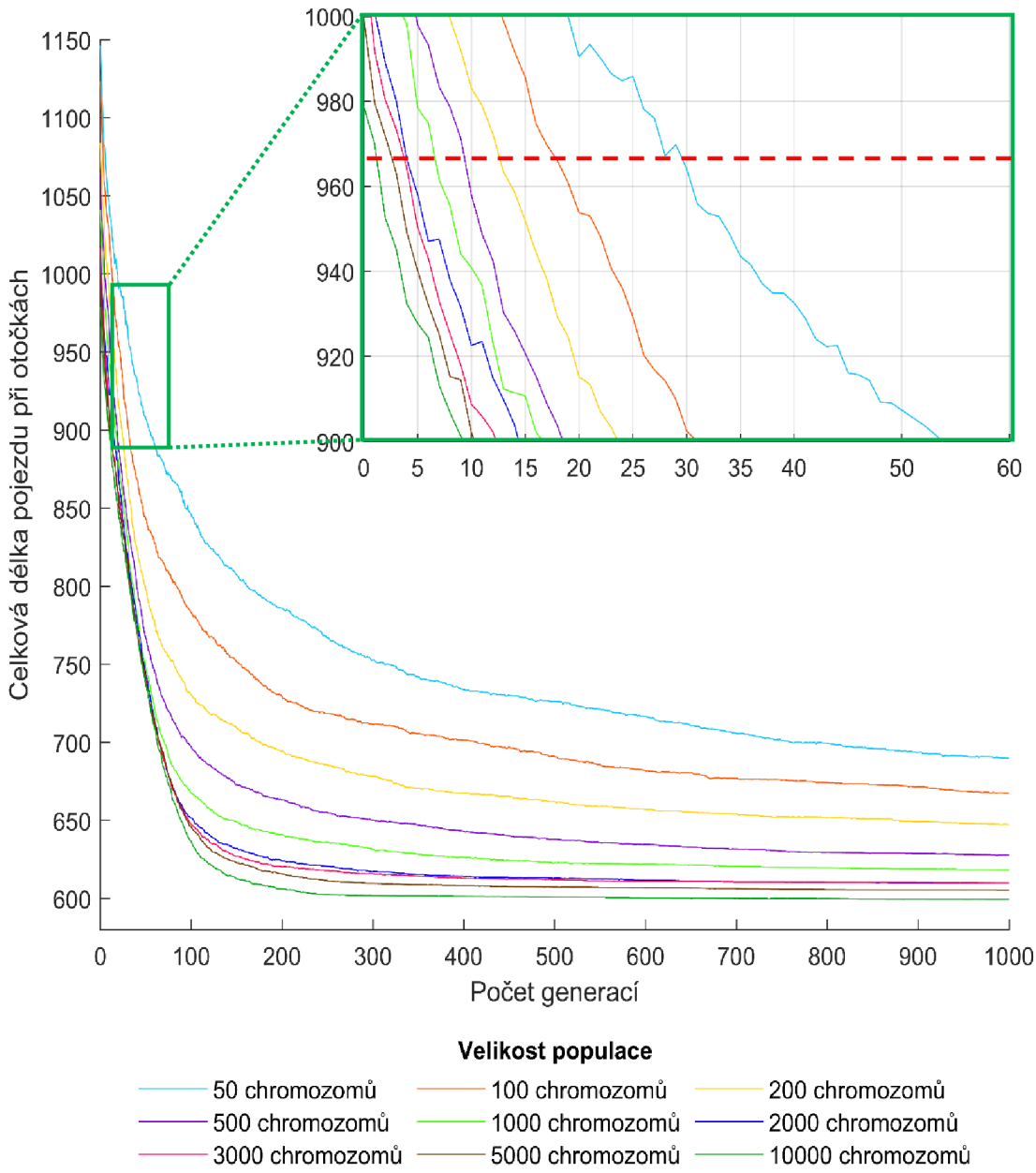
Obrázek 4.1: Znázornění vygenerovaného testovacího pozemku s trajektorií členkového pojezdu

4.1 Analýza výsledků z hlediska vlivu velikosti populace

Pro sledování vlivu velikosti populace a počtu generací byla zvolena pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 a posuvná mutace s pravděpodobností 0,2. Výpočty probíhaly až do 1000 generací, přičemž velikost populace charakterizovaná počtem chromozomů, kdy jeden chromozom představuje možné řešení průjezdu dílčích tras, nabývá hodnot 50, 100, 200, 500, 1000, 2000, 3000, 5000, 10000. Z grafu 4.1 zobrazujícího minimální celkovou délku tras při otáčení, je patrné, že během generací dochází k nalezení optimálnějších řešení s menší hodnotou délky pojezdu u populací všech testovaných velikostí, k nejrychlejší konvergenci vzhledem k minimální hodnotě v generaci 1000 dochází u populace s 10000 chromozomy, jenž dosahuje také nejlepšího řešení z testovaných velikostí populací, nejhorších výsledků dosahuje populace nejmenší. Nejstrmější průběh křivky lze sledovat v rozmezí generací 0 až 200 napříč všemi uvedenými počty chromozomů, například v rámci prvních 100 epoch klesla hodnota u největší populace o více než 30 % a při tisíci chromozomech téměř o 36 %. Ve výřezu oblasti grafu 4.1 o intervalu $\langle 900; 1000 \rangle$ na ose y a $\langle 0; 60 \rangle$ na ose x červenou přerušovanou čárou vynesena celková délka otoček při článkovém pohybu techniky, která pro zkoumaný pozemek činní 966,747 m, tuto hodnotu překonávají řešení s počtem 500-10000 chromozomů již před generací 10 a nejmenší populace dosahuje stabilně lepších výsledků od 30. epochy.



Obrázek 4.2: Znázornění vygenerovaného testovacího pozemku s optimální trajektorií získanou v rámci populace velikosti 10000 v generaci 1000



Graf 4.1: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro vybrané velikosti populace, selekce pořadová, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

Tabulka 4.1 ukazuje hodnoty minimální celkové délky tras při otáčení ve vybraných generacích, kdy s rostoucím počtem chromozomů dochází k získání lepších výsledků, což může být spojeno s vyšší diverzitou u větších populací vedoucí k možnému zvýšení pravděpodobnosti nalezení lepšího řešení. Vygenerované výsledky se zlepšují také s počtem generací, což odpovídá principu fungování genetických algoritmů. Nejvyšší hodnoty, 1146,27 m, což je 118,57 % délky člunkového pojezdu viz ta-

bulka 4.2, dosahuje populace s 50 chromozomy v generaci 0, naopak největší populace dosáhla nejnižší hodnoty, 599,49 m, v poslední generaci. Z hlediska rozdílu hodnot v generaci 1000 je největší rozdíl u menších populací, kdy velikostně po sobě jdoucí populace mají rozdíl pohybující se okolo 3 % vůči populaci předchozí, u populace 1000 a 2000 chromozomů dochází ke snížení rozdílu k 1,5 %, mezi generacemi 2000 a 3000 je rozdíl nepatrný a následně se pohybuje v rozmezí 0,5-1 %.

Tabulka 4.1: Minimální celková délka tras při otáčení v průběhu vybraných generací z rozsahu 0 až 1000 pro vybrané velikosti populace, selekce pořadová, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

	Generace								
P_{CH}[1]	**0**	**100**	**200**	**300**	**400**	**500**	**600**	**800**	**1000**
50	1146,27	845,95	786,04	753,02	733,92	726,36	716,63	699,66	689,76
100	1123,36	783,26	729,32	711,36	701,41	690,81	681,73	674,27	667,08
200	1083,85	730,63	695,00	678,34	667,61	662,04	657,21	652,00	647,40
500	1063,76	697,17	663,22	650,07	643,18	637,92	634,60	629,54	627,95
1000	1041,19	667,68	640,99	631,36	626,50	623,10	622,11	619,76	618,47
2000	1017,24	651,64	624,13	617,18	614,35	613,08	611,76	610,34	609,84
3000	1016,90	647,40	620,31	615,97	613,06	612,16	611,00	610,40	609,87
5000	999,85	646,06	615,76	609,76	608,31	607,37	606,90	605,65	605,36
10000	978,98	636,74	606,18	601,93	601,47	600,92	600,30	599,98	599,49
Čl. p.[2]	966,747								

V porovnání s článkovým pojezdem nedochází k nalezení lepšího řešení během nulté generace v žádné populaci, avšak v generaci 100 je tento způsob pojezdu překonán o více než 12 % u nejmenší populace a o téměř 35 % u populace největší, úplně nejlepší řešení vykazuje v poslední generaci o 37,99 % menší délku pojezdu. Z tabulky 4.2 je rovněž zjevné, že úsporu 30 % a více zajistí všechny populace obsahující alespoň 100 chromozomů, přičemž jejich počet má důležitý vliv na počet epoch potřebných k dosažení této hodnoty, kdy se u větších populací objevují lepší hodnoty při nižším počtu generací než u populací s menším počtem chromozomů.

[1] Počet chromozomů v populaci
[2] Článkový pojezd

Tabulka 4.2: Poměr minimální celkové délky tras při otáčení v průběhu generací 0 až 1000 pro vybrané velikosti populace vůči délce otoček při člunkovém pojezdu [%]

	Generace								
P_{CH}	0	100	200	300	400	500	600	800	1000
50	118,57	87,51	81,31	77,89	75,92	75,13	74,13	72,37	71,35
100	116,20	81,02	75,44	73,58	72,55	71,46	70,52	69,75	69,00
200	112,11	75,58	71,89	70,17	69,06	68,48	67,98	67,44	66,97
500	110,04	72,11	68,60	67,24	66,53	65,99	65,64	65,12	64,96
1000	107,70	69,06	66,30	65,31	64,81	64,45	64,35	64,11	63,97
2000	105,22	67,41	64,56	63,84	63,55	63,42	63,28	63,13	63,08
3000	105,19	66,97	64,16	63,72	63,42	63,32	63,20	63,14	63,09
5000	103,42	66,83	63,69	63,07	62,92	62,83	62,78	62,65	62,62
10000	101,27	65,86	62,70	62,26	62,22	62,16	62,09	62,06	62,01

S rostoucí velikostí populace je spojena i zvyšující se časová náročnost výpočtů, což je přímo spojeno s vyšším počtem provedených operací. Nejmenší průměrný čas potřebný pro průběh 100 generací je 1, 71 s při skupině řešení o 50 chromozomech, největší skupina potřebuje pro vygenerování stejného počtu generací 382,83 s, trend růstu potřeby výpočetního času je úměrný také množství provedených epoch, viz tabulka 4.3.

Tabulka 4.3: Celkový čas potřebný k výpočtu minimální délky pojezdových tras při otáčení pro zvolený počet generací a velikost populace [s]

	Generace								
P_{CH}	0	100	200	300	400	500	600	800	1000
50	0,0170	1,71	3,41	5,11	6,80	8,50	10,20	13,59	16,99
100	0,0322	3,25	6,48	9,70	12,92	16,14	19,36	25,81	32,25
200	0,0640	6,46	12,86	19,25	25,65	32,05	38,44	51,24	64,03
500	0,1559	15,74	31,33	46,92	62,51	78,10	93,69	124,86	156,04
1000	0,3048	30,78	61,26	91,74	122,22	152,70	183,18	244,13	305,09
2000	0,5587	56,43	112,30	168,17	224,04	279,91	335,78	447,51	559,25
3000	0,9282	93,75	186,58	279,40	372,22	465,05	557,87	743,52	929,17
5000	1,5045	151,96	302,41	452,86	603,31	753,77	904,22	1205,12	1506,03
10000	3,7904	382,83	761,88	1140,92	1519,96	1899,00	2278,05	3036,13	3794,22

Vzhledem ke značným rozdílům výpočetního času napříč populacemi i generacemi a méně enormní diverzity minimálních vzdáleností je třeba zvolit optimální nastavení

parametrů nalezení vyváženosti mezi kvalitou výsledků a časovou náročností. Tabulky 4.4 a 4.5 znázorňují bodové ohodnocení optimální kombinace počtu generací a velikosti populace s ohledem na výpočetní čas a minimální vzdálenost pojezdu při stanovených váhách. Posouzení jednotlivých variací bylo provedeno na základě propojení tabulky 4.2 a tabulky 4.3 převedené do procentuálních hodnot, kdy je hodnota buňky $b_{m,n}$ v nové tabulce na místě m, n rovna součtu součinu hodnoty buňky $t_{m,n}$ z upravené tabulky 4.3 a váhy u a součinu $s_{m,n}$ z tabulky 4.2 a váhy $100 - u$, viz vzorec 4.1

$$b_{m,n} = t_{m,n}u + s_{m,n}(100 - u), u \in \langle 0; 100 \rangle. \quad (4.1)$$

Pokud je při výpočtu dána priorita době jeho trvání, tak $u > 50$, v případě přiřazení vyššího důrazu kvalitě výsledků nabývá váha u hodnot nižších, nejoptimálnější jsou konfigurace s nejnižší hodnotou v tabulce. Z tabulky 4.4 vyplývá, že i při vyšší potřebě krátkého výpočetního času, kdy poměr vah 90 ku 10 ve prospěch času, není nejrychlejší varianta optimální, ale nejvhodnější volbou je populace o počtu 200 chromozomů za průběhu 200-400 generací. Při rovnovážném stavu, tedy poměru vah 50: 50 vychází nejlépe kombinace 2000 chromozomů a 200 generací, případně 1000 chromozomů a počet generací opět z rozsahu 200-400, viz tabulka 4.5. Je zajímavé, že se v této situaci stále jeví nejrychlejší řešení jako podstatně vhodnější možnost než nejlepší řešení. Dle tabulky 4.6, kde je kladen důraz především na získání vysoké kvality řešení s nízkým ohledem na čas v poměru, se nachází nejvíce optimální nastavení také při volbě 2000 chromozomů se zpracováním po 300-600 epoch.

Tabulka 4.4: Bodové ohodnocení znázorňující optimální konfigurace počtu generací a velikosti populace pro váhové ohodnocení 90:10

	Generace							
P_{CH}	**100**	**200**	**300**	**400**	**500**	**600**	**800**	**1000**
50	8,79	8,21	7,91	7,75	7,72	7,65	7,56	7,53
100	8,18	7,70	7,59	7,56	7,53	7,51	7,59	7,64
200	7,71	7,49	7,47	7,51	7,61	7,71	7,96	8,08
500	7,58	7,60	7,84	8,14	8,45	8,79	9,47	9,84
1000	7,64	8,08	8,71	9,38	10,07	10,78	12,20	12,92
2000	8,08	9,12	10,37	11,67	12,98	14,29	16,93	18,25
3000	8,92	10,84	13,00	15,17	17,36	19,55	23,95	26,15
5000	10,29	13,54	17,05	20,60	24,16	27,73	34,85	38,42
10000	15,67	24,34	33,29	42,28	51,26	60,25	78,22	87,21

Tabulka 4.5: Bodové ohodnocení znázorňující optimální konfigurace počtu generací a velikosti populace pro váhové ohodnocení 50:50

	Generace							
P_{CH}	100	200	300	400	500	600	800	1000
50	43,78	40,70	39,01	38,05	37,68	37,20	36,37	36,06
100	40,55	37,81	36,92	36,45	35,94	35,51	35,21	35,12
200	37,87	36,11	35,34	34,87	34,66	34,50	34,40	34,33
500	36,26	34,71	34,24	34,09	34,02	34,06	34,21	34,37
1000	34,94	33,96	33,86	34,01	34,24	34,59	35,27	35,63
2000	34,45	33,76	34,14	34,73	35,40	36,07	37,46	38,18
3000	34,72	34,54	35,54	36,61	37,79	38,95	41,37	42,59
5000	35,42	35,83	37,50	39,41	41,35	43,30	47,21	49,18
10000	37,98	41,39	46,17	51,14	56,10	61,07	71,04	76,02

Tabulka 4.6: Bodové ohodnocení znázorňující optimální konfigurace počtu generací a velikosti populace pro váhové ohodnocení 10:90

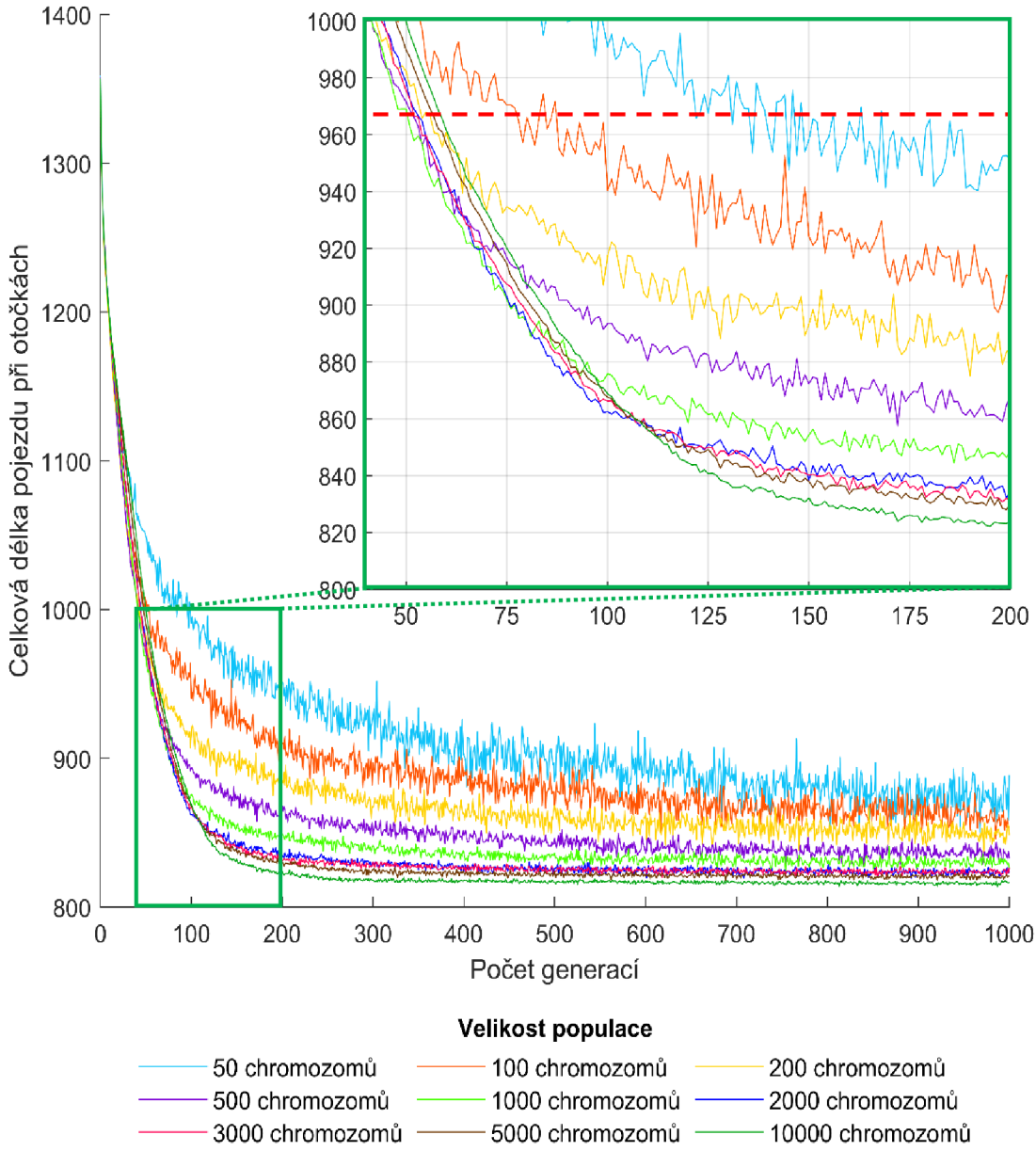
	Generace							
P_{CH}	100	200	300	400	500	600	800	1000
50	78,76	73,19	70,12	68,34	67,64	66,74	65,17	64,59
100	72,93	67,91	66,25	65,33	64,35	63,52	62,84	62,60
200	68,04	64,74	63,20	62,22	61,72	61,28	60,83	60,58
500	64,94	61,83	60,64	60,04	59,59	59,33	58,94	58,91
1000	62,24	59,83	59,02	58,65	58,41	58,40	58,34	58,34
2000	60,81	58,40	57,90	57,78	57,81	57,84	58,00	58,10
3000	60,52	58,24	58,08	58,05	58,22	58,35	58,79	59,02
5000	60,55	58,12	57,96	58,22	58,53	58,88	59,56	59,95
10000	60,29	58,44	59,04	60,00	60,95	61,89	63,86	64,82

Průměrné hodnoty všech řešení v rámci jednotlivých generací kopírují trend křivek popisujících minimální vzdálenosti včetně charakteru strmého poklesu v intervalu generace 0 až 200. V rámci jednotlivých epoch však dochází k výkyvům hodnot, které se značně zvyšují se snižujícím počtem chromozomů v populaci, kdy například směrodatná odchylka a variační koeficient během posledních 10 epoch je u největší populace 0,58 a 0,071 %, kdežto u populace nejmenší již 10,58 a 1,21 %, tedy hodnoty více než o 1600 % vyšší. Z výřezu v grafu 4.2 vyplývá, že i průměrné hodnoty u populací s 200 a více chromozomy překonávají článkový pojezd ještě před dosažením

75 generace, dále je zajímavé, že do uplynutí 100 epoch podávají populace s 2000-5000 chromozomy lepší výsledky než soubor 10000 chromozomů, od generace 111 se však tato situace mění. V tabulce 4.7 lze stejně jako tabulce 4.1 pozorovat pozitivní vliv rostoucího počtu generací i chromozomů na kvalitu průměrného řešení, avšak v tomto případě v generaci nula jsou rozdíly napříč populacemi zanedbatelné, jejich výrazné navýšení je patrné až v generaci 100.

Tabulka 4.7: Průměrná celková délka tras při otáčení v průběhu vybraných generací z rozsahu 0 až 1000 pro vybrané velikosti populace, selekce pořadová, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

	Generace								
P_{CH}	0	100	200	300	400	500	600	800	1000
50	1358,97	990,83	950,80	928,94	901,17	884,49	904,85	877,75	895,19
100	1352,45	940,30	909,96	897,15	888,50	865,73	872,99	865,14	866,13
200	1356,08	917,49	885,18	867,11	862,77	868,44	863,26	859,47	852,47
500	1357,81	893,51	868,04	847,97	848,08	846,15	836,05	836,82	834,97
1000	1357,21	875,84	849,01	841,32	836,78	836,35	831,75	836,05	829,48
2000	1356,57	862,42	834,98	830,43	823,31	826,14	825,85	821,84	824,88
3000	1356,56	866,25	831,17	830,27	825,35	824,16	826,37	823,50	824,94
5000	1356,53	867,61	830,03	825,39	822,83	824,21	820,44	818,94	820,90
10000	1357,20	868,75	824,21	817,95	818,29	816,61	817,89	816,78	816,77



Graf 4.2: Průměrná celková délka tras při otáčení v průběhu generací 0 až 1000 pro vybrané velikosti populace, selekce pořadová, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

4.2 Analýza výsledků s ohledem na vliv typu selekční metody

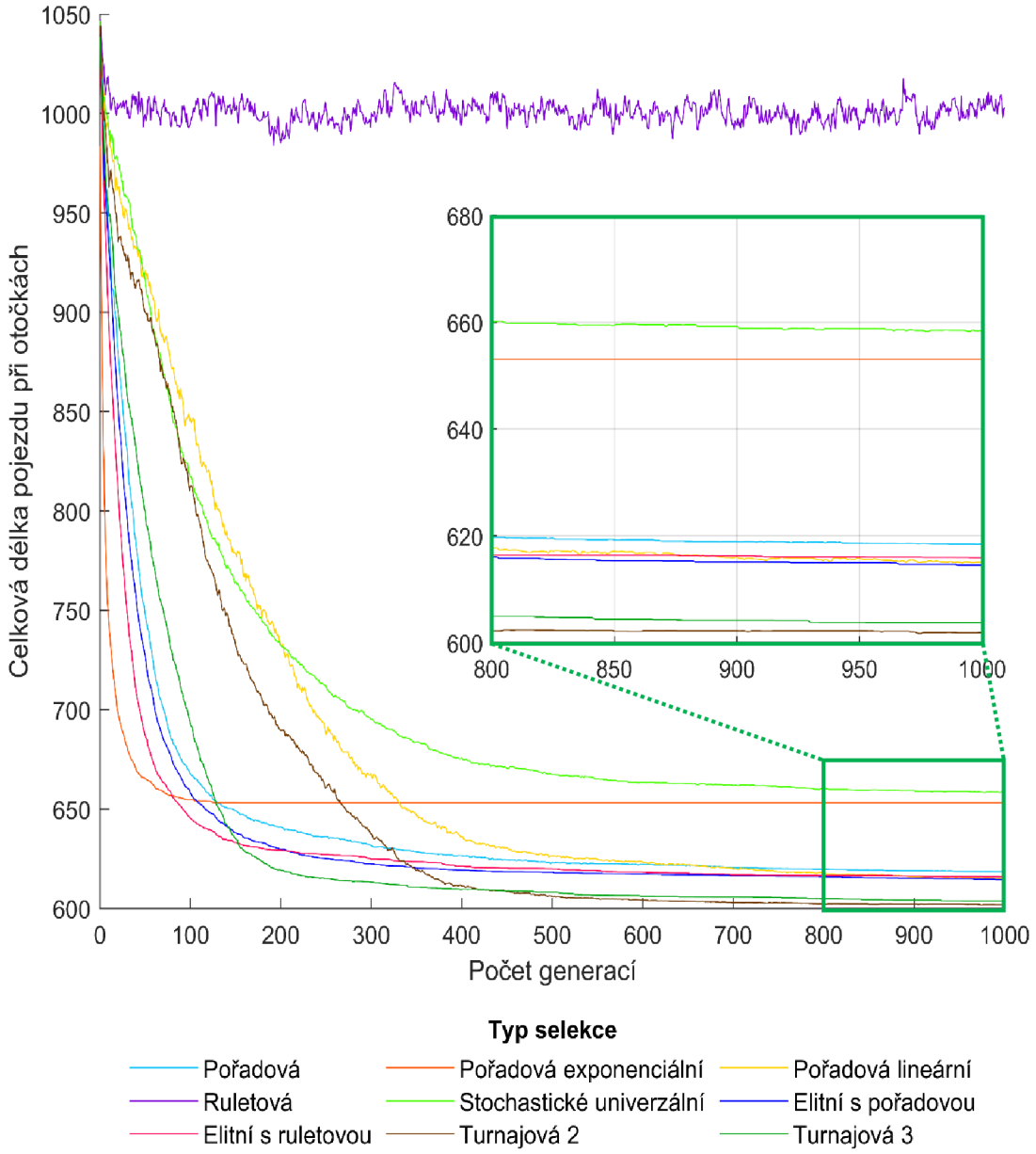
Data pro porovnání selekčních metod byla získána při nastavení velikosti populace 1000 chromozomů, křížení na základě pozice s pravděpodobností 0,9 a posuvné mutace s pravděpodobností 0,2, celý proces probíhal po 1000 generací. Nejlepších výsledků dosahovala turnajová selekce v obou variantách, viz graf 4.3, přičemž při výběru porovnáním dvou chromozomů konvergovala pomaleji než při porovnání tří, například v generaci 200 vykazovala o 11,33 % vyšší hodnoty, avšak v generaci 400

činil rozdíl nepatrných 0,31 % a v poslední epoše již vygenerovala 0,31 % lepší řešení. Selekce pořadová, elitní v kombinaci s pořadovou i ruletovou a pořadová lineární vykazovaly taktéž obstojné výsledky lišící se v generaci 1000 od turnajové metody o 2,10-2,76 %, ale průběh pořadové lineární selekce byl až do generace 500 rozdílný z hlediska rychlosti poklesu hodnot, kdy ještě v generaci 200 má nalezená minimální trasa o 14,20 % větší délku než v případě selekce pořadové. Pořadová exponenciální má zajímavý průběh během prvních 50 generací, kdy překonává ostatní testované typy, výrazný pokles hodnot se však rychle zpomaluje a od epochy 126 je zcela zastaven, což je pravděpodobně dáno vysokou hodnotou základu exponentu 0,9 z intervalu $(0; 1)$, v případě volby nižšího základu by měl být očekávaný průběh křivky méně strmý. Ruletová selekce se neprokázala jako vhodná volba pro řešenou problematiku, dosahovala jednoznačně nejhorších výsledků, které ani po uplynutí 1000 generací nepřekonaly článkový pojezd, což může být zapříčiněno menšími rozdíly po ohodnocení mezi jednotlivými chromozomy vstupní populace, čímž může docházet k náhodnému výběru nevhodných řešení. Tento problém je u ostatních typů selekce ošetřen, například nahrazením hodnoty fitness hodnotou pořadí, přičemž jsou i menší rozdíly po ohodnocení pořadím zvýrazněny, nebo výběrem části nejlepších chromozomů.

Tabulka 4.8: Minimální celková délka tras při otáčení v průběhu vybraných generací z rozsahu 0 až 1000 pro vybrané typy selekce, velikost populace 1000, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

	Generace								
Typ selekce	**0**	**100**	**200**	**300**	**400**	**500**	**600**	**800**	**1000**
Pořadová	1041,19	667,68	640,99	631,36	626,50	623,10	622,11	619,76	618,47
Pořadová exponenciální	983,55	654,67	653,12	653,12	653,12	653,12	653,12	653,12	653,12
Pořadová lineární	1040,47	846,70	732,02	667,03	636,24	626,45	623,14	617,74	615,10
Ruletová	1047,24	994,71	985,26	1000,28	995,37	1002,84	998,83	991,73	1002,96
Stochastická univerzální	1046,89	817,07	732,24	695,22	675,54	667,46	663,37	660,24	658,41
Elitní s pořadovou	1040,61	657,85	629,79	622,27	619,26	618,22	617,26	616,06	614,52
Elitní s ruletovou	1036,59	645,38	629,28	624,83	621,24	619,97	618,22	616,40	615,92

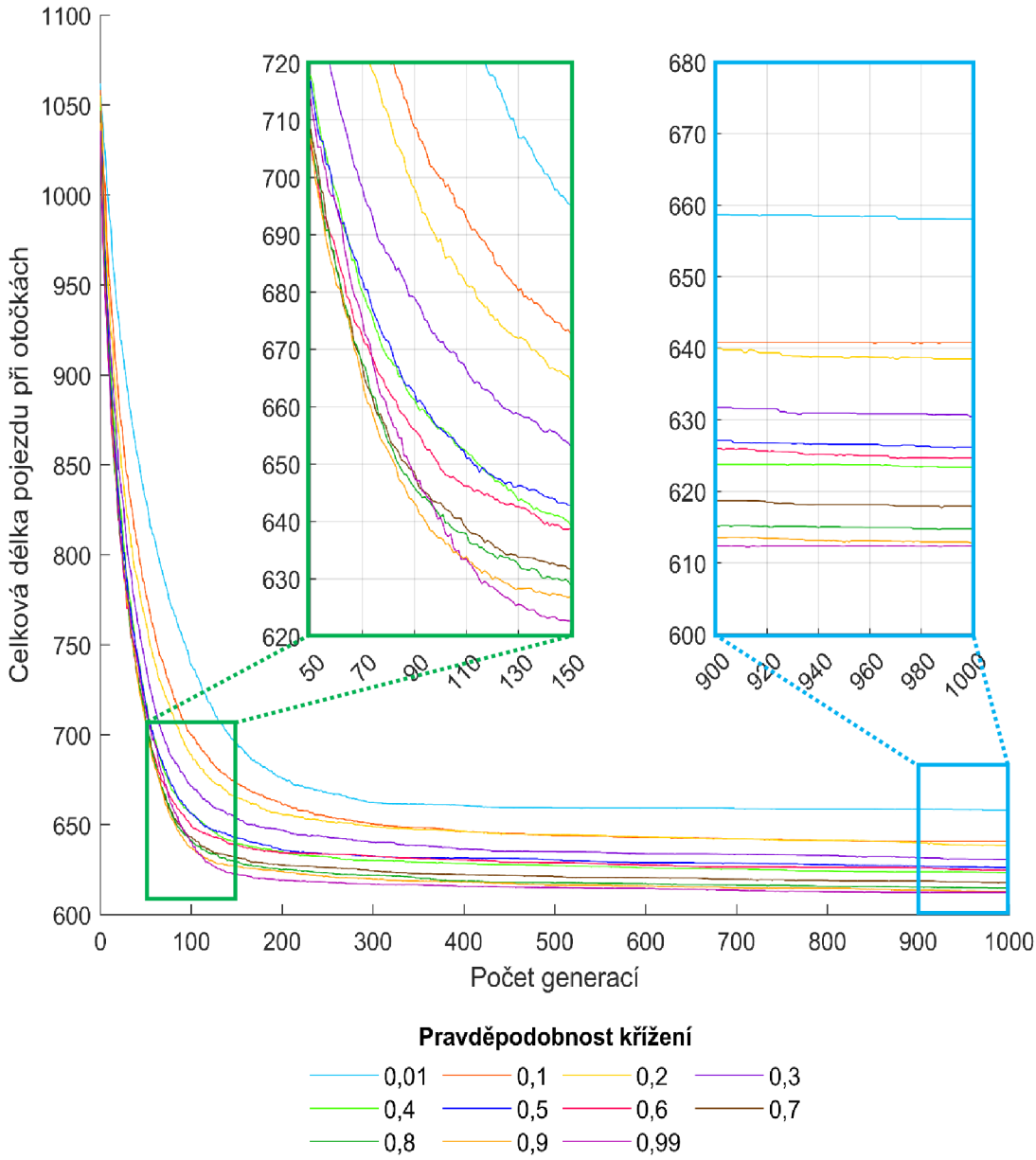
Turnajová s výběrem 2 jedinců	1044,21	812,38	689,39	637,51	611,39	606,10	604,26	602,17	601,88
Turnajová s výběrem 3 jedinců	1038,48	693,12	619,23	613,11	609,50	608,12	606,06	604,89	603,76



Graf 4.3: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro vybrané typy selekce, velikost populace 1000, křížení na základě pozice s pravděpodobností 0,9, posuvná mutace s pravděpodobností 0,2 [m]

4.3 Analýza výsledků s ohledem na vliv typu a pravděpodobnosti křížení

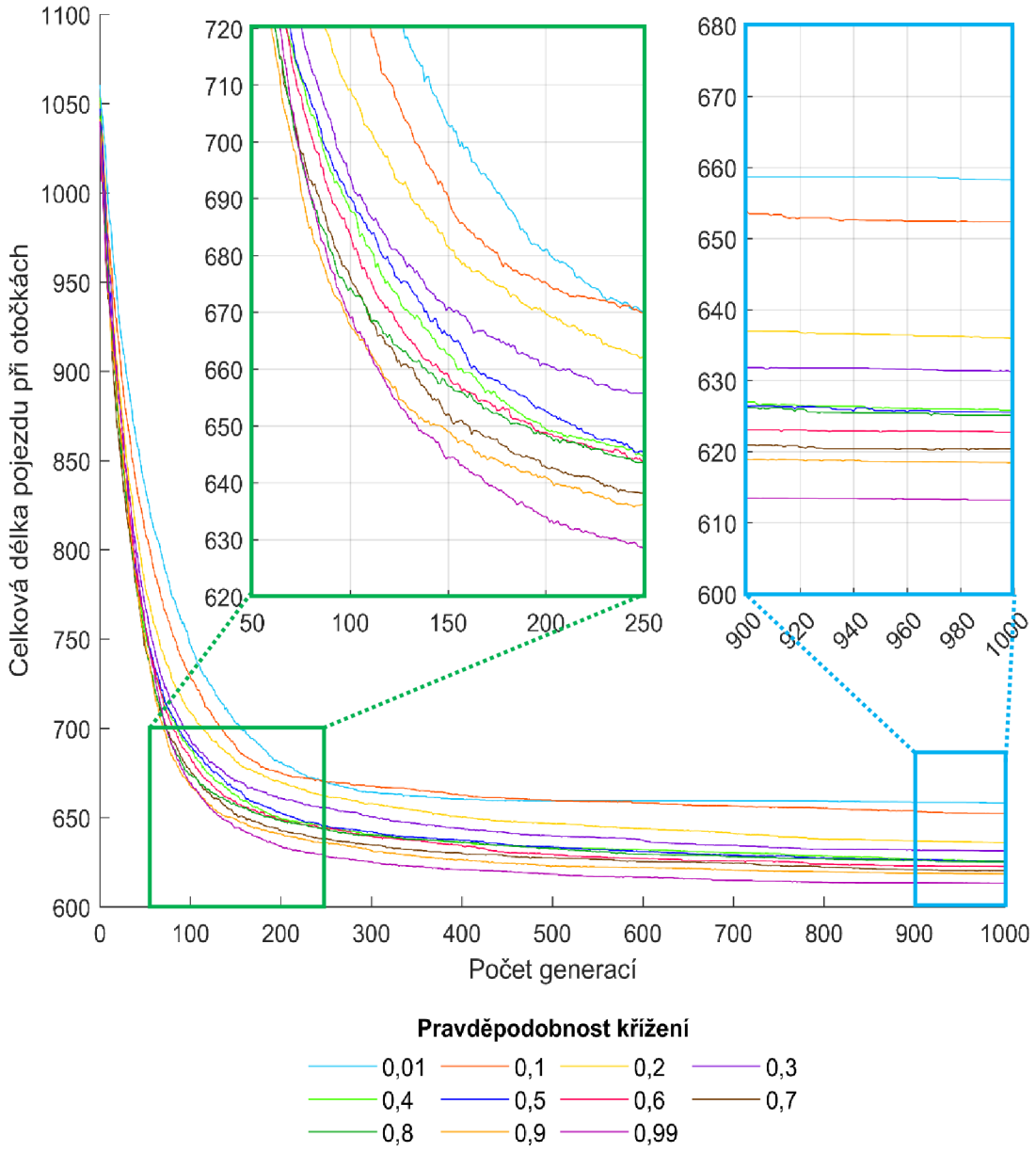
Pro porovnání vybraných typů křížení a ověření vztahu mezi velikostí pravděpodobnosti křížení P_c a kvalitou výsledných řešení byla data vygenerována pro populaci 1000 chromozomů, pořadovou selekci a posuvnou mutací s pravděpodobností 0,2.



Graf 4.4: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro křížení se zachováním pořadí realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, posuvná mutace s pravděpodobností 0,2 [m]

Z grafů 4.4 a 4.5 je jasně patrný trend zlepšujících se výsledků v průběhu generací, který je výraznější se zvyšující se pravděpodobností křížení u typu na základě pořadí

i na základě pozice, avšak první typ dosahuje nižších hodnot téměř při každé pravděpodobnosti v průběhu všech generací, například pro $P_c = 0{,}9$ v poslední generaci má minimální hodnotu 612,73 m oproti 618,47 m. Z grafu 4.4 dále vyplývá, že pro nízký počet generací při aplikaci křížení se zachováním pořadí je vhodnější zvolit druhou největší pravděpodobnost křížení, a to až do generace 110, poté je již při $P_c = 0{,}99$ dosaženo nižších hodnot, stejná situace nastává i u křížení na základě pozice, viz graf 4.5.



Graf 4.5: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro křížení na základě pozice realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, posuvná mutace s pravděpodobností 0,2 [m]

Tabulka 4.9: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro křížení na základě pořadí realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, posuvná mutace s pravděpodobností 0,2 [m]

	Generace								
P_c[3]	0	100	200	300	400	500	600	800	1000
0,01	1061,90	738,54	675,95	662,30	660,54	659,39	659,36	658,73	658,08
0,10	1058,55	699,83	661,19	650,40	646,20	643,80	642,95	641,14	640,77
0,20	1055,27	689,28	655,63	648,94	646,32	644,51	643,20	641,20	638,53
0,30	1045,29	671,27	646,95	640,13	636,49	634,99	633,78	633,01	630,44
0,40	1048,86	656,45	635,01	630,10	628,45	627,24	625,88	624,35	623,36
0,50	1039,78	656,20	635,88	632,25	631,55	630,38	628,87	627,78	626,15
0,60	1028,25	649,17	634,41	632,45	630,39	628,87	627,55	626,25	624,65
0,70	1047,02	643,29	627,52	624,45	622,24	621,08	620,56	619,17	617,91
0,80	1039,60	640,81	625,01	621,82	618,68	617,61	617,28	615,65	614,75
0,90	1039,89	637,15	623,93	619,58	618,19	616,47	615,77	614,57	612,73
0,99	1035,76	640,66	619,30	616,86	615,68	615,08	614,41	612,79	612,35

Tabulka 4.10: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro křížení na základě pozice realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, posuvná mutace s pravděpodobností 0,2 [m]

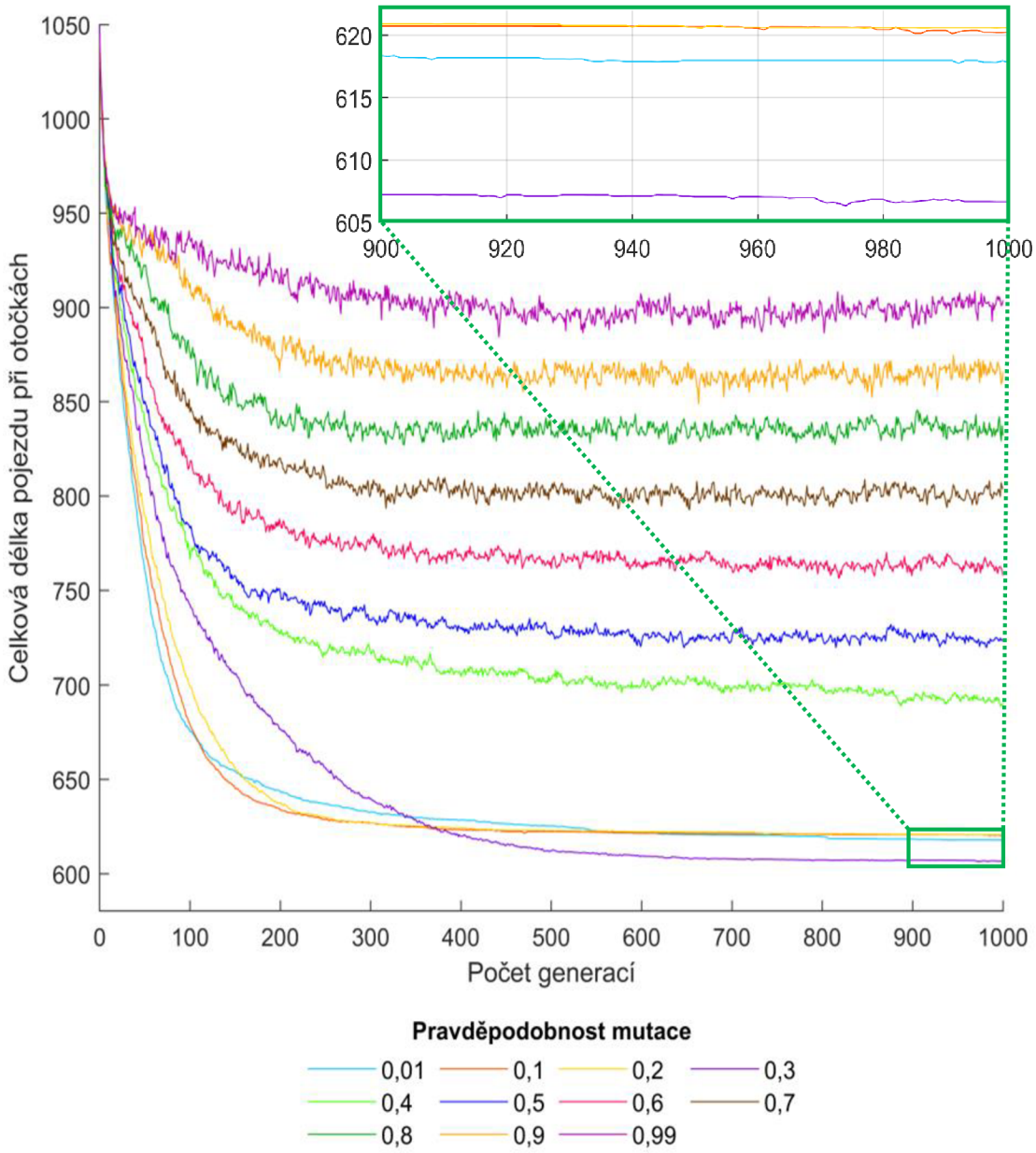
	Generace								
P_c	0	100	200	300	400	500	600	800	1000
0,01	1060,44	746,67	681,19	664,30	660,41	659,49	659,49	659,49	659,29
0,10	1050,03	728,88	675,23	667,80	662,47	659,58	658,27	656,53	655,35
0,20	1051,33	709,37	669,66	657,91	650,06	646,56	643,78	641,19	637,96
0,30	1052,19	693,82	660,82	650,43	643,78	639,52	637,44	634,52	632,33
0,40	1053,55	687,88	649,34	640,54	635,85	633,31	632,05	630,48	628,48
0,50	1047,20	690,23	652,54	642,29	637,48	633,62	631,03	628,99	627,58
0,60	1037,47	683,66	648,85	639,51	634,42	629,43	627,18	625,75	624,04
0,70	1039,85	676,06	642,72	634,83	630,05	627,51	625,43	624,29	622,33
0,80	1043,10	674,31	648,48	640,06	635,94	632,60	629,51	628,24	627,06

[3] Pravděpodobnost křížení

0,90	1041,19	667,68	640,99	631,36	626,50	623,10	622,11	620,57	619,76
0,99	1039,90	668,92	634,10	625,28	620,94	618,36	616,75	614,84	613,69

4.4 Analýza výsledků s ohledem na vliv typu a pravděpodobnosti mutace

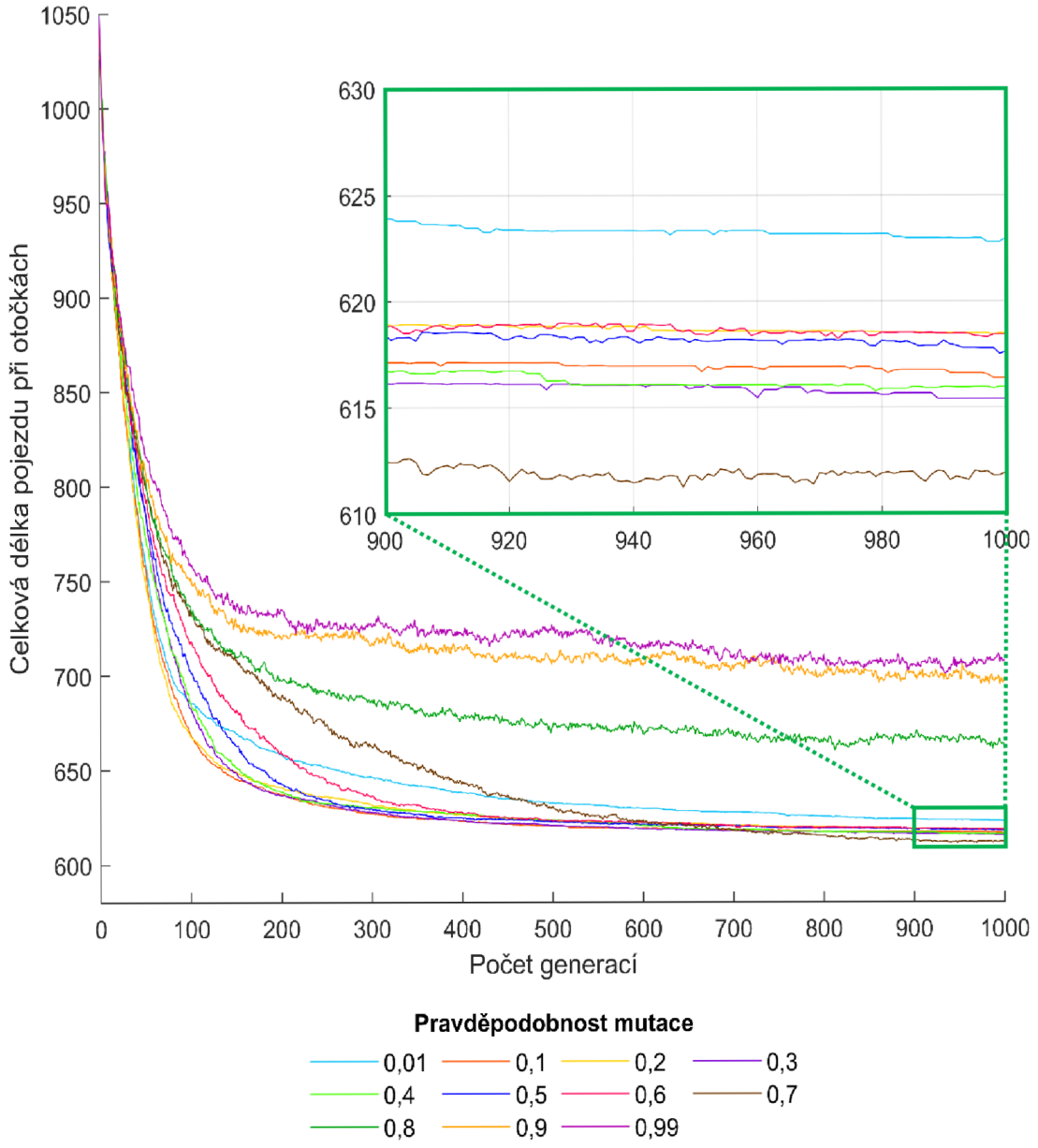
Pro porovnání vybraných typů mutace a ověření vztahu mezi velikostí pravděpodobnosti mutace P_m a kvalitou výsledných řešení byla data vygenerována pro populaci 1000 chromozomů, pořadovou selekci a křížení na základě pozice s pravděpodobností 0,9. Mutace výměnou vykazuje vysokou citlivost na velikost pravděpodobnosti mutace, kdy do $P_m = 0{,}3$ jsou získány velmi dobré výsledky, přičemž při této hodnotě dochází k dosažení minima z vygenerovaných dat, od $P_m = 0{,}4$ se kvalita výsledků začíná snižovat, tedy s dále rostoucí pravděpodobností roste také celková délka pojezdu při otočkách v rámci získaných řešení. Do prvních 100 epoch se jeví jako nejoptimálnější volba nejmenší testovaná pravděpodobnost, poté však rychlost její konvergence klesá a lepších výsledků dosahuje $P_m = 0{,}1$. Křivka $P_m = 0{,}3$ klesá zpočátku pomaleji, ale trend jejího vývoje se nemění tak razantně, pokles je pozvolnějšího a trvalejšího rázu, viz graf 4.6, a tedy od generace 366 již poskytuje nejlepší hodnoty, což může být zapříčiněno i tím, že pomocí křížení již nelze po několik stech generacích zajistit větší heterogenitu optimálních chromozomů populace a nízká pravděpodobnost mutace tento vývoj ovlivňuje nedostatečně, kdežto u pravděpodobnosti mutace 0,3 umožní v rozumné míře získání nových variant optimálních řešení. Avšak vyšší P_m již mění část optimálních chromozomů v populaci výrazněji a ve výsledku zhoršuje jejich kvalitu.



Graf 4.6: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro mutaci výměnou realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]

Při aplikaci posuvné mutace není negativní vliv vysoké pravděpodobnosti natolik výrazný jako u mutace výměnou, při $p_m = 0{,}99$ jsou získané hodnoty o 21,68 % nižší, u $P_m = 0{,}8$ činní rozdíl 20,57 %. Tento jev je pravděpodobně dán odlišným principem mutace, kdy při výměně genů v chromozomu dochází k podstatnějším změnám v rámci kombinace průjezdu dílčích tras než při posunu pořadí vybrané části celého chromozomu o jedno místo. Z grafu 4.7 je dále patrné, že nejrychleji konverguje

křivka pro pravděpodobnosti s hodnotami 0,1 a 0,2, avšak od generace 700 dále vykazuje nejlepší výsledky $P_m = 0{,}7$. V generaci 1000 nejsou u pravděpodobností z rozsahu 0,1 až 0,6 rozdíly hodnot nijak zásadní, nejnižší a nejvyšší z této skupiny se liší pouze o 0,5 %.



Graf 4.7: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro posuvnou mutaci realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]

Tabulka 4.11: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro mutaci výměnou realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]

	Generace								
P_M[4]	0	100	200	300	400	500	600	800	1000
0,01	1038,21	675,26	643,51	632,76	628,48	625,15	621,47	619,54	617,83
0,10	1027,76	679,09	633,98	626,86	623,24	622,36	621,47	620,86	620,23
0,20	1041,40	699,22	637,67	626,79	624,40	622,96	622,25	621,24	620,62
0,30	1048,37	742,35	676,80	639,80	619,99	612,31	609,31	607,37	606,64
0,40	1043,42	766,35	727,66	716,58	704,58	703,07	700,31	698,13	690,63
0,50	1044,02	782,94	748,88	734,68	729,72	728,84	722,86	725,35	723,43
0,60	1032,09	818,38	785,36	776,24	768,81	761,55	762,98	763,10	757,87
0,70	1038,08	844,40	820,13	801,23	802,85	803,92	803,71	804,17	803,61
0,80	1038,41	875,50	840,37	835,21	830,98	832,37	836,70	838,00	832,17
0,90	1044,44	912,63	871,99	866,40	864,73	860,63	870,66	869,74	865,77
0,99	1044,88	935,62	913,72	906,47	905,62	895,52	905,25	896,19	903,78

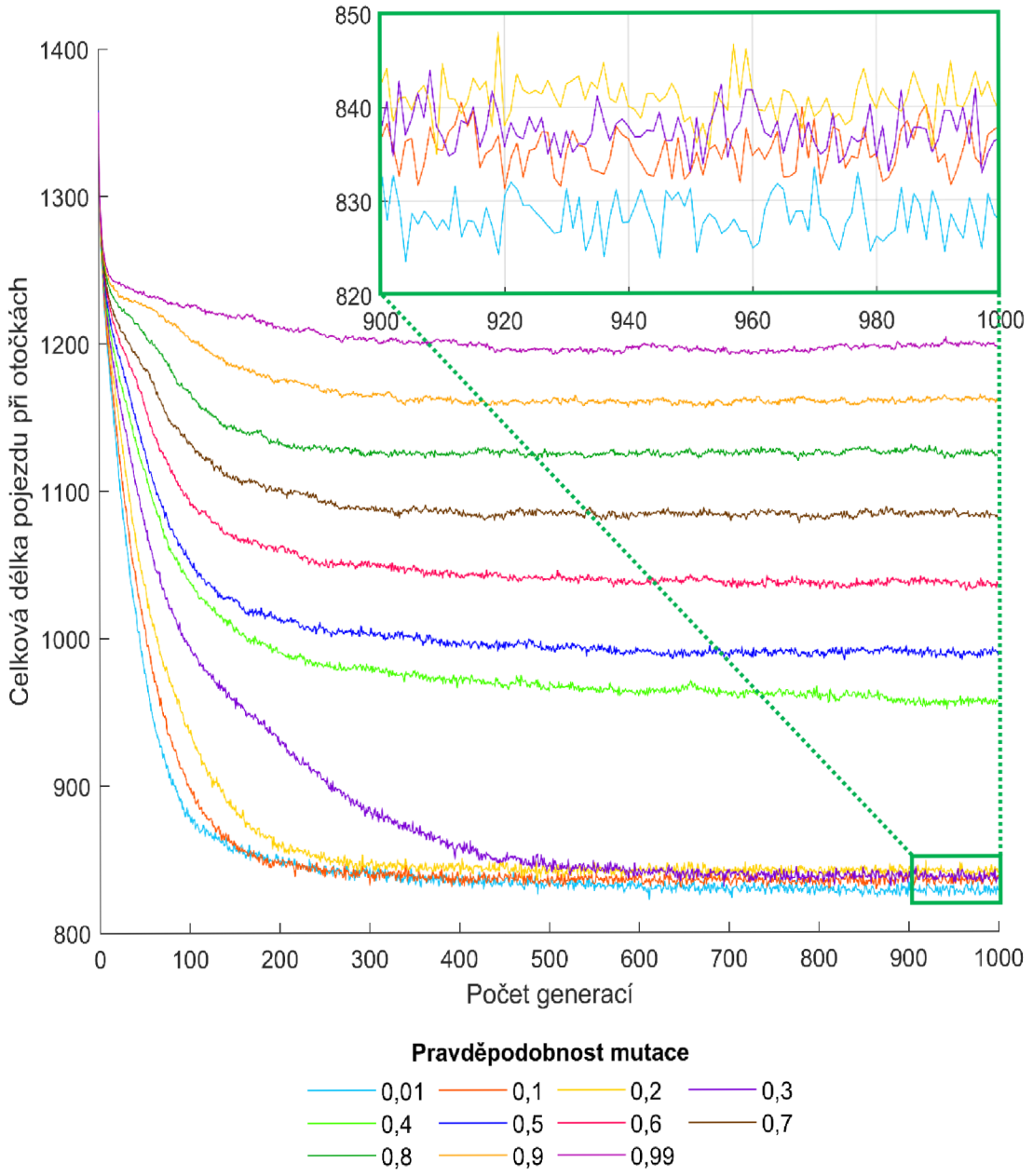
Na základě získaných dat lze konstatovat, že vhodná volba velikosti pravděpodobnosti křížení je pro vygenerování optimálních řešení klíčová, v případě příliš vysoké pravděpodobnosti bude docházet v populaci po selekci a křížení ke změnám u velkého množství chromozomů, což není s ohledem na možné znehodnocení značné části optimálních řešení žádoucí. Nízká pravděpodobnost závratně neovlivní celkový vývoj, ale může zvýšit heterogenitu populaci bez negativního vlivu na většinu chromozomů. Posuvná mutace je více odolná proti vysoké P_m než mutace výměnou a ve většině z testovaných variant dochází k lepším výsledkům, výjimku však tvoří P_m o velikosti 0,01 a 0,3, kdy dokonce u druhé zmíněné došlo k dosažení celkové nejnižší hodnoty délky otoček 606,64 m. Tabulky 4.11 a 4.12 obsahují hodnoty minimální celková délka tras při otáčení v průběhu vybraných generací.

[4] Pravděpodobnost mutace

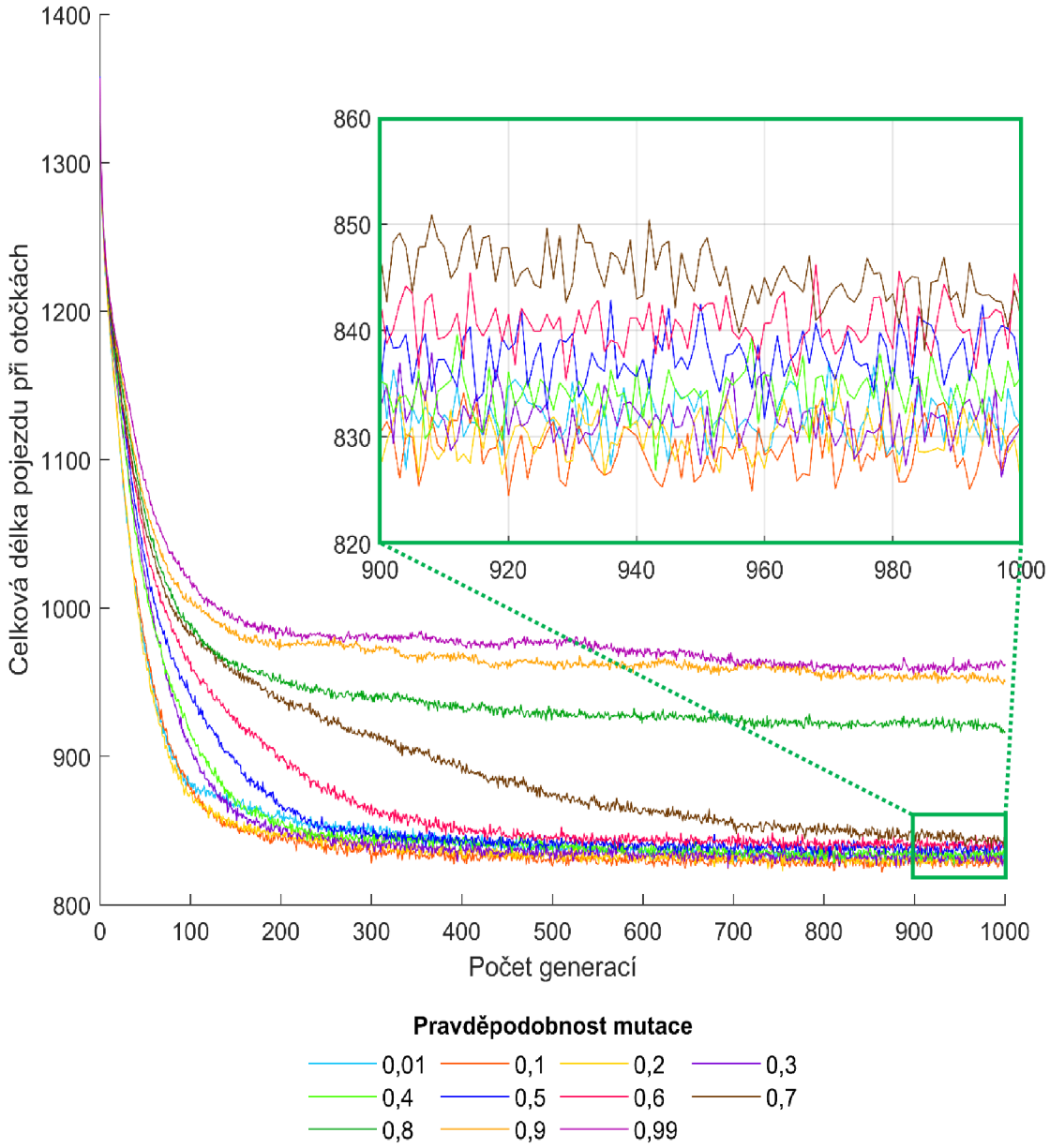
Tabulka 4.12: Minimální celková délka tras při otáčení v průběhu generací 0 až 1000 pro posuvnou mutaci realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]

	Generace								
P_M	0	100	200	300	400	500	600	800	1000
0,01	1038,21	685,58	658,29	646,25	638,03	632,59	629,52	625,30	622,99
0,10	1027,49	668,43	637,17	626,89	623,08	620,38	619,11	617,33	616,39
0,20	1041,19	667,68	640,99	631,36	626,50	623,10	622,11	619,76	618,47
0,30	1048,93	682,45	636,78	627,53	623,35	620,64	618,74	617,19	615,42
0,40	1045,47	684,49	638,32	630,32	626,96	623,58	620,65	617,08	615,97
0,50	1043,31	701,47	642,59	629,41	624,73	622,65	621,24	619,08	617,62
0,60	1037,49	718,13	658,84	636,61	627,31	623,07	622,40	619,34	618,43
0,70	1040,93	731,66	689,12	662,38	643,79	630,69	623,34	615,26	611,95
0,80	1039,60	734,01	697,30	687,03	677,92	673,00	668,98	666,69	660,96
0,90	1044,01	751,78	719,37	717,25	713,44	708,15	705,13	701,49	698,35
0,99	1038,21	685,58	658,29	646,25	638,03	632,59	629,52	625,30	622,99

Vývoj hodnot průměrné celkové délky tras otáčení je podobný jako u minimálních hodnot pro oba typy mutace, u mutace výměnou dochází pro pravděpodobnost 0,4 a větší k poklesu kvality výsledků, při zvolení nižších hodnot jsou výsledky výrazně lepší, přičemž s jejich poklesem roste rychlost konvergence, průměrně nejlepší řešení nastávají při výběru $P_m = 0{,}01$. Výkyvy mezi hodnotami v průběhu generací nejsou napříč testovanými pravděpodobnostmi výrazně odlišné, směrodatná odchylka a variační koeficient během posledních 100 epoch se pohybují v rozmezí 1,14-2,46 a 0,098-0,271 %. V případě posuvné mutace je situace obdobná, nejlepších hodnot je dosaženo při pravděpodobnosti do 0,4, na rozdíl od minimální celkové vzdálenosti již $P_m = 0{,}7$ v generaci 1000 ostatní varianty nepředčí, což však může být dáno pozvolnějším průběhem klesání hodnot. Opět jsou zde zjevné drobné výkyvy, směrodatná odchylka a variační koeficient během posledních 100 epoch se pohybují v rozmezí 1,87-2,33 a 0,197-0,308 %.



Graf 4.8: Průměrná celková délka tras při otáčení v průběhu generací 0 až 1000 pro mutaci výměnou realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]



Graf 4.9: Průměrná celková délka tras při otáčení v průběhu generací 0 až 1000 pro posuvnou mutaci realizované s vybranými pravděpodobnostmi, velikost populace 1000, pořadová selekce, křížení na základě pozice s pravděpodobností 0,9 [m]

4.5 Analýza výsledků s ohledem na vliv počtu dílčích pojezdových tras a generací

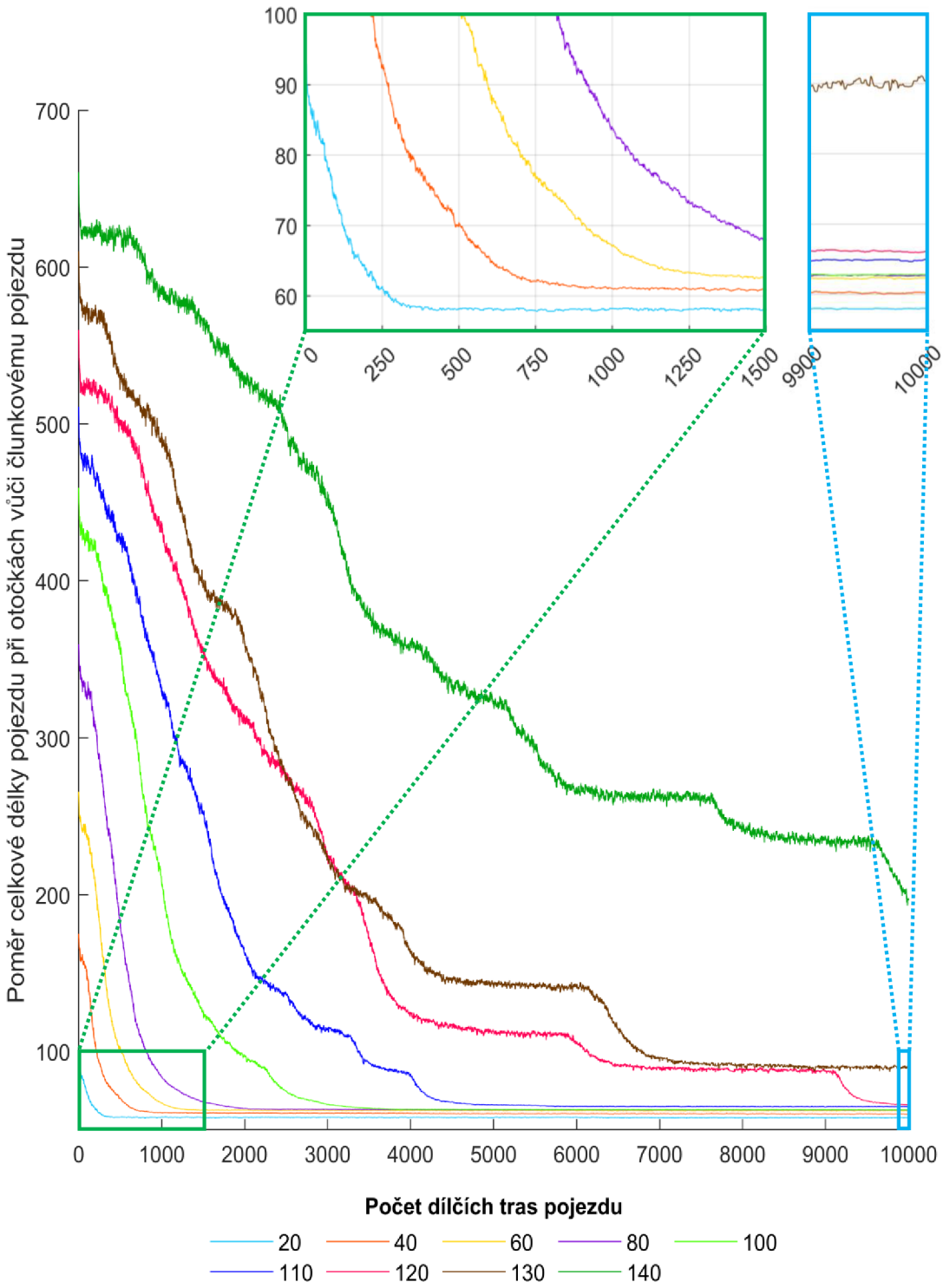
Pro tento pokus byl vygenerován nový testovací pozemek obdélníkového tvaru o rozměrech 130×200 m, u něhož lze snadno škálovat počet dílčích pojezdových tras změnou jeho velikosti. Pro všechny varianty bylo zvoleno stejné sestavení parametrů, kdy poloměru otáčení i záběru byla přiřazena hodnota 5 m, velikost populace činila 500 chromozomů, zvolena byla turnajová selekce s porovnáním dvou jedinců, křížení

na základě pozice s pravděpodobností 0,99 a posuvná mutace s pravděpodobností 0,1, testování probíhalo po 10000 generací. Průběh algoritmu byl pro každou velikost pozemku realizován dvacetkrát, získané výsledky byly zprůměrovány. V tabulce 4.13 je ukázán počet možných kombinací průjezdu jednotlivých tras v závislosti na jejich počet a čas potřebný pro průběh algoritmu v rámci stanovený počet epoch.

Tabulka 4.13: Počet možných kombinací průjezdu tras v závislosti na jejich počtu

Násobek základní velikosti pozemku	Počet tras	Počet možných kombinací průjezdu jednotlivých tras	Doba jednoho průběhu algoritmu [s]
1x	20	2,4329E+18	878,5659
2x	40	8,15915E+47	1747,157
3x	60	8,32099E+81	2597,474
4x	80	7,1569E+118	3529,845
5x	100	9,3326E+157	4285,406
5.5x	110	1,5882E+178	4850,397
6x	120	6,6895E+198	5430,455
6.5x	130	6,4669E+219	6017,708
7x	140	1,3462E+241	6237,412

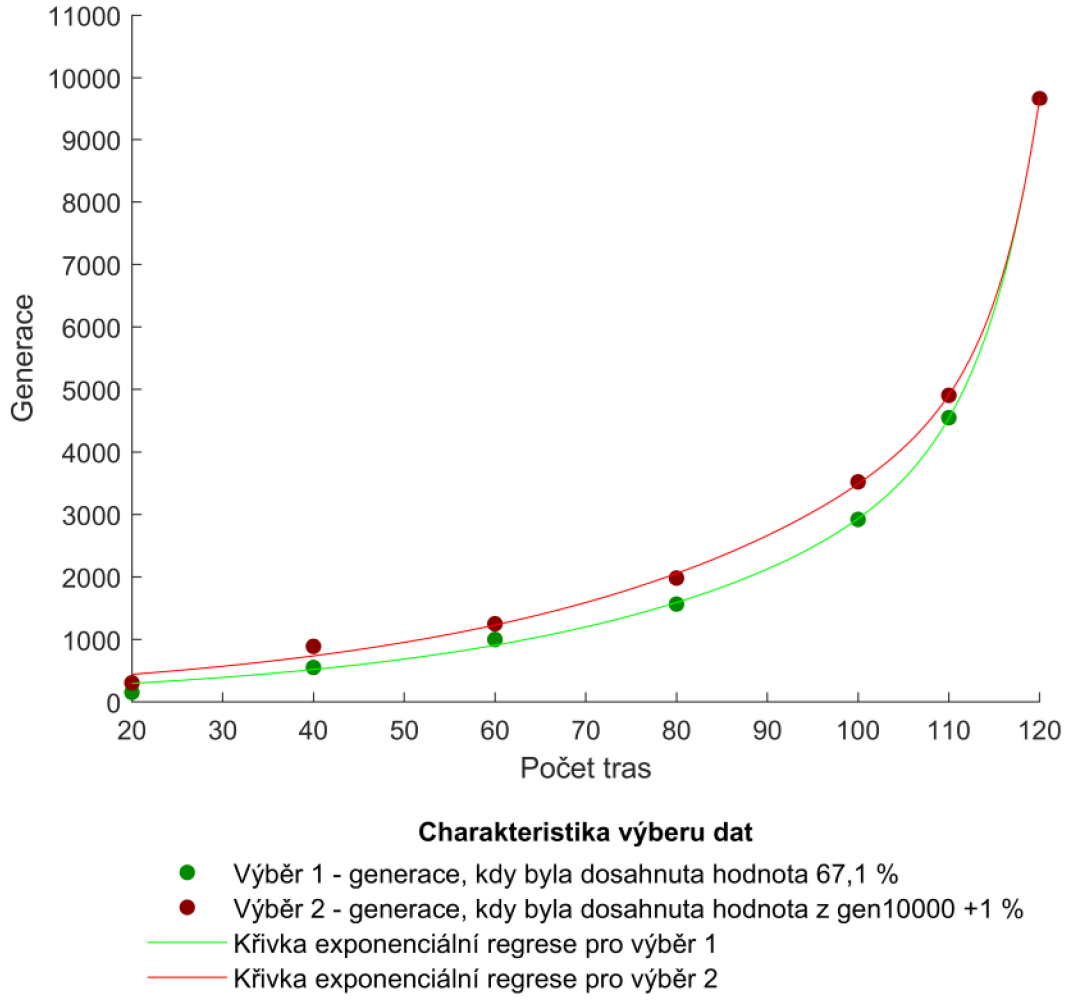
S rostoucím počtem pojezdových tras rapidně klesá kvalita výsledků získaných během prvních generací, kdy pro první velikost pozemku v generaci 100 dosahovala vygenerovaná řešení 73,6 % hodnoty délky člunkového pojezdu, velikost pozemku 2 již 145,8 % a například výsledky páté varianty vykazovaly 430 %. U všech testovaných možností je prokazatelný klesající trend, avšak odlišný je jeho průběh, s rostoucím počtem tras se zvyšuje také množství generací potřebných pro dosažení poklesu rychlosti konvergence, kdy je strmost křivky hodnot pozvolnější a pokles je vůči testovanému množství setrvalejšího rázu. Se zvyšující se četností tras se také objevují oblasti ustrnutí v lokálním minimu, tento jev nabývá na výraznosti od 110 tras, přičemž je zvláště patrný u 120 a více tras, viz graf 4.10.



Graf 4.10: Poměr minimální celkové délky tras při otáčení v průběhu generací 0 až 10000 vůči délce člunkového pojezdu pro vybraný počet pojezdových tras, velikost populace 500, turnajová selekce s porovnáním dvou jedinců, křížení na základě pozice s pravděpodobností 0,1 [%]

Pro získání optimálních výsledků je klíčový počet generací, který se odvíjí od velikosti pozemku, potažmo množství dílčích tras, s jehož zvyšováním roste počet potřebných generací exponenciálně, viz graf 4.11, ve kterém jsou zaneseny počty generací, které

bylo třeba realizovat v prvním případě pro získání hodnot menších než 67,1 % člunkového pojezdu a v druhém pro hodnoty o pouhé 1% větší než v generaci 10 000, následně byly výsledky proloženy exponenciální regresní křivkou.



Graf 4.11: Vliv počtu pojezdových tras na množství generací potřebných pro získání optimálních výsledků

Značně menší vzdálenost pojezdu lze pozorovat u variant do 120 tras, kdy dochází k jejímu snížení minimálně o 33,9 % v případě šestinásobné velikosti pozemku až o 42 % u nejmenší varianty, viz tabulka 4.14. Při 130 trasách již výsledky nevykazují významnou úsporu, rozdíl je pouze 10 %, pravděpodobně by stejně jako u největší varianty bylo třeba počet generací markantně navýšit.

Tabulka 4.14: Poměr minimální celkové délky tras při otáčení v průběhu generací 0 až 10000 vůči délce článkového pojezdu pro vybraný počet pojezdových tras, velikost populace 500, turnajová selekce s porovnáním dvou jedinců, křížení na základě pozice s pravděpodobností 0,1 [%]

	Generace								
P_{pt}[5]	0	1000	2000	3000	4000	5000	6000	8000	10000
20	94,46	58,21	58,08	58,16	58,00	58,00	58,13	57,88	58,00
40	175,08	61,01	60,68	60,53	60,63	60,37	60,42	60,49	60,28
60	265,67	66,98	62,70	62,52	62,54	62,46	62,35	62,27	62,32
80	359,61	83,84	63,65	63,12	62,95	62,84	62,79	62,78	62,69
100	459,37	207,02	97,41	66,15	63,14	63,06	62,91	62,94	62,79
110	510,83	330,13	161,99	114,73	85,73	65,86	65,09	64,89	64,96
120	559,87	435,24	311,84	223,85	125,12	112,29	106,37	86,28	66,09
130	610,14	490,63	359,64	224,87	163,75	143,55	140,38	91,33	90,00
140	660,47	585,24	530,56	455,67	361,89	323,89	267,72	245,30	196,45

Tabulka 4.15: Minimální celková délka tras při otáčení v průběhu generací 0 až 10000 pro vybraný počet pojezdových tras, velikost populace 500, turnajová selekce s porovnáním dvou jedinců, křížení na základě pozice s pravděpodobností 0,99, posuvná mutace s pravděpodobností 0,1 [m]

	Generace					Článkový pojezd
P_{pt}	0	1000	2000	3000	4000	
20	541,33	333,62	332,87	333,32	332,37	573,09
40	2059,51	717,72	713,75	712,02	713,25	1176,34
60	4727,77	1191,96	1115,72	1112,62	1112,97	1779,60
80	8568,86	1997,75	1516,79	1504,01	1500,09	2382,85
100	13717,18	6181,81	2908,83	1975,33	1885,31	2986,10
110	16794,65	10853,63	5325,83	3771,86	2818,50	3287,73
120	20095,59	15622,28	11193,04	8034,76	4490,83	3589,36
130	23740,47	19090,46	13993,50	8749,56	6371,39	3890,98
140	27690,82	24537,03	22244,21	19104,27	15172,79	4192,61

[5] Počet pojezdových tras

Tabulka 4.16: Minimální celková délka tras při otáčení v průběhu generací 5000 až 10000 pro vybraný počet pojezdových tras, velikost populace 500, turnajová selekce s porovnáním dvou jedinců, křížení na základě pozice s pravděpodobností 0,99, posuvná mutace s pravděpodobností 0,1 [m]

	Generace				Článkový pojezd
P_{pt}	5000	6000	8000	10000	
20	332,37	333,12	331,87	331,68	573,09
40	710,11	710,80	708,41	711,55	1176,34
60	1111,53	1109,53	1110,00	1108,11	1779,60
80	1497,48	1496,12	1496,23	1495,93	2382,85
100	1882,89	1878,58	1879,08	1879,31	2986,10
110	2165,20	2140,07	2131,77	2133,52	3287,73
120	4030,43	3817,98	3178,72	3096,77	3589,36
130	5585,36	5462,30	3646,50	3553,81	3890,98
140	13579,33	11224,34	11161,98	10284,57	4192,61

Závěr

Aplikací genetických algoritmů při optimalizaci pojezdových tras zemědělské techniky lze dosáhnout výrazného zkrácení délky otoček realizovaných při přejezdech mezi dílčími drahami ve vnitřní části pole, kdy může dojít ke snížení vzdálenosti i o více než 37 %, což může přímo vést k úsporám pracovního času, pohonných hmot a nižšímu zhutnění půdy na souvratích.

Značný vliv na kvalitu nalezeného řešení má velikost populace chromozomů, tedy skupina možných řešení, i počet generací během nichž dochází evolučním úpravám zlepšujícím výsledky, přičemž se zvyšováním těchto parametrů dochází ke generování optimálnějších řešení, avšak s tím je spojen i růst výpočetního času. Během testování implementovaných algoritmů se ukázalo, že volba nejvyšších hodnot velikosti populace i počtu provedených epoch není vždy nejvhodnější, protože získání o několik procent kratší trasy je za cenu značného navýšení času simulace. Výběr selekční metody může také do kvality výsledků zasáhnout, například ruletová selekce se prokázala jako zcela nevhodná pro využití v této problematice, naopak turnajový či pořadový typ vykazovaly velmi dobré hodnoty. Při aplikaci křížení byl patrný menší vliv typu metody než velikosti pravděpodobnosti, přičemž s jejím růstem docházelo ke zlepšování získaných výsledků. U mutace je třeba zvolit hodnotu pravděpodobnosti s ohledem na vybranou metodu, protože se její vliv liší. Mutaci výměnou je vhodné použít s nižší pravděpodobností, dle realizovaných testů do velikosti 0,3, naopak posuvná mutace vykazuje optimální výsledky až do pravděpodobnosti 0,7. Nezanedbatelný dopad na získání vhodných řešení má také počet dílčích pojezdových tras, při jehož zvýšení kvalita výsledků během prvních sto generací rapidně klesá. Zároveň dochází k exponenciálnímu zvyšování množství generací potřebných pro dosažení adekvátních výsledků, což je pravděpodobně spojeno s intenzivním růstem počtu možných kombinací průjezdů jednotlivých tras.

Genetické algoritmy prokázaly svůj význam při řešení problematiky pohybu zemědělských strojů po pozemku, avšak oblast metaheuristických algoritmů je rozsáhlá a zahrnuje velké množství optimalizačních metod, které mají v zemědělství široké použití, z nichž lze mnohé navrhnout pro implementaci do stávajícího modelu.

Seznam použité literatury

Aladeemy, M., Adwan, L., Booth, A., Khasawneh, M.T., Poranki, S. (2020). New feature selection methods based on opposition-based learning and self-adaptive cohort intelligence for predicting patient no-shows. Applied Soft Computing, 86. https://doi.org/10.1016/j.asoc.2019.105866.

Alaiso, S., Backman, J., & Visala, A. (2013). Ant Colony Optimization for Scheduling of Agricultural Contracting Work. Ifac Proceedings Volumes , 46(18). https://doi.org/10.3182/20130828-2-SF-3019.00041

Ali, H., Gong, D., Wang, M., & Dai, X. (2020). Path Planning of Mobile Robot With Improved Ant Colony Algorithm and MDP to Produce Smooth Trajectory in Grid-Based Environment. Frontiers In Neurorobotics, 14. https://doi.org/10.3389/fnbot.2020.00044

Alipour, M. M. et al. (2018). A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. Neural Computing and Applications. Springer London, 30(9). https://doi.10.1007/s00521-017-2880-4.

Amal, L., Son, L. H. and Chabchoub, H. (2018). SGA: spatial GIS-based genetic algorithm for route optimization of municipal solid waste collection. Environmental Science and Pollution Research. Environmental Science and Pollution Research, 25(27). https://doi.10.1007/s11356-018-2826-0.

Assaf, R. and Saleh, Y. (2017). Vehicle-Routing Optimization for Municipal Solid Waste Collection Using Genetic Algorithm: The Case of Southern Nablus City. Civil and Environmental Engineering Reports, 26(3). https://doi.10.1515/ceer-2017-0034.

Aydemir, E. and Karagul, K. (2020). Solving a Periodic Capacitated Vehicle Routing Problem Using Simulated Annealing Algorithm for a Manufacturing Company. Braz. J. Oper. Prod. Manag., 17(1). https://doi.org/10.14488/bjopm.2020.011.

Bakhtiari, A., Navid, H., Mehri, J., Berruto, R., Bochtis, D.D. (2013). Operations planning for agricultural harvesters using ant colony optimization. Span. J. Agric. Res., 11(3). https://doi.org/10.5424/sjar/2013113-3865

Baniamerian, A., Bashiri, M. and Tavakkoli-Moghaddam, R. (2019). Modified variable neighborhood search and genetic algorithm for profitable heterogeneous

vehicle routing problem with cross-docking. Applied Soft Computing. Elsevier B.V., 75. https://doi.org/10.1016/j.asoc.2018.11.029

Barrientos, A., Colorado, J., Cerro, J. del, Martinez A., Rossi, C., Sanz, D., Valente, J. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. J. Field Robot., 28(5). https://doi.org/10.1002/rob.20403

Beasley, J. E. and Chu, P. C. (1996). A genetic algorithm for the set covering problem. European Journal of Operational Research, 94(2). https://doi.org/10.1016/0377 2217(95)00159-X.

Behnck, L. P., Doering, D., Pereira, C. E., & Rettberg, A. (2015). A Modified Simulated Annealing Algorithm for SUAVs Path Planning. Ifac-Papersonline, 48(10). https://doi.org/10.1016/j.ifacol.2015.08.109

Bochtis, D.D. et Vougioukas, S.G. (2008). Minimising the non-working distance travelled by machines operating in a headland field pattern. Biosyst. Eng., 101(1). https://doi.org/10.1016/j.biosystemseng.2008.06.008

Bulgakov, V., Pascuzz, S., Nadykto, V., Adamchuk, V., Kaminskiy, V., Kyurchev, V., Santoro, F. (2022). Effects of Tractor and Soil Parameters on the Depth of the Permanent Traffic Lanes in Controlled Traffic Farming Systems. Applied Sciences., 12(13). https://doi.org/10.3390/app12136620

Cárceles Rodríguez B, Durán-Zuazo VH, Soriano Rodríguez M, García-Tejero IF, Gálvez Ruiz B, Cuadros Tavira S. (2022). Conservation Agriculture as a Sustainable System for Soil Health: A Review. Soil Systems, 6(4). https://doi.org/10.3390/soilsystems6040087

Centeri C. (2022). Effects of Grazing on Water Erosion, Compaction and Infiltration on Grasslands. Hydrology, 9(2). https://doi.org/10.3390/hydrology9020034

Cerdeira-Pena, A., Carpente, L., & Amiama, C. (2017). Optimised forage harvester routes as solutions to a traveling salesman problem with clusters and time windows. Biosyst. Eng., 164. https://doi.org/10.1016/j.biosystemseng.2017.10.002

Conesa-Muñoz, J. et al. (2016). Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications. Computers and Electronics in Agriculture., 127. https://doi.org/10.1016/j.compag.2016.06.012

Dao, S. D., Abhary, K. and Marian, R. (2017). A bibliometric analysis of Genetic Algorithms throughout the history. Computers and Industrial Engineering. Elsevier Ltd, 110. https://doi.10.1016/j.cie.2017.06.009.

Darwish, A.; Hassanien, A.E.; Das, S. (2020). A survey of swarm and evolutionary computing approaches for deep learning. Artif. Intell. Rev., 53. https://doi.org/10.1007/s10462-019-09719-2

Das, P. K. et Jena, P. K. (2020). Multi-robot path planning using improved Particle Swarm Optimization algorithm through novel evolutionary operators. Appl. Soft Comput. J. Elsevier B.V., 92. https://doi.org/10.1016/j.asoc.2020.106312.

De León-Aldaco, S.E., Calleja, H., Aguayo Alquicira, J. (2015). Metaheuristic Optimization Methods Applied to Power Converters: A Review. IEEE Transactions on Power Electronics, 30(12). https://doi.org/10.1109/TPEL.2015.2397311.

De Lima, R.P., da Silva, A.P., Giarola, N.F.B, da Silva, A.R., Rolim, M.M. (2017). Changes in soil compaction indicators in response to agricultural field traffic. Biosystems Engineering, 162. https://doi.org/10.1016/j.biosystemseng.2017.07.002.

Dhiman, G. et Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Adv. Eng. Softw., 114. https://doi.org/10.1016/j.advengsoft.2017.05.014

Edwards, G.T.C, Hinge, J., Skou-Nielsen, N., Villa-Henriksen, A., Sørensen, C.A.G, Green, O. (2017). Route planning evaluation of a prototype optimised infield route planner for neutral material flow agricultural operations. Biosystems Engineering, 153. https://doi.org/10.1016/j.biosystemseng.2016.10.007.

Elferchichi, A. et al. (2009). The genetic algorithm approach for identifying the optimal operation of a multi-reservoirs on-demand irrigation system. Biosystems Engineering. IAgrE, 102(3). https://doi.10.1016/j.biosystemseng.2008.12.009.

Elhoseny, M., Tharwat, A. and Hassanien, A. E. (2018). Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm. Journal of Computational Science. Elsevier B.V., 25. https://doi.10.1016/j.jocs.2017.08.004.

Elshaer, R., Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering, 140. https://doi.org/10.1016/j.cie.2019.106242.

Feng, Z. (2020). Constructing rural e-commerce logistics model based on ant colony algorithm and artificial intelligence method. Soft Comput. Springer Berlin Heidelberg, 24(11). https://doi.org/10.1007/s00500-019-04046-8.

Ferreira Neto, J. A. et al. (2011). Optimal subdivision of land in agrarian reform projects: an analysis using genetic algorithms. Ciencia e investigación agraria, 38(2). https://doi.10.4067/rcia.v38i2.111.

Francik, S., Knapczyk, A., Wójcik, A., Ślipek, Z. (2020). Optimisation Methods in Renewable Energy Sources Systems—Current Research Trends. In: Wróbel, M., Jewiarz, M., Szlęk , A. (eds) Renewable Energy Sources: Engineering, Technology, Innovation. Springer Proceedings in Energy. Springer, Cham. https://doi.org/10.1007/978-3-030-13888-2_81

Gao, M. et al. (2017). An Improved Genetic Algorithm for Island Route Planning. Procedia Engineering. Elsevier B.V., 174(1). https://doi.org/10.1016/j.proeng.2017.01.163.

Glover, F. (1989). Tabu Search—Part I. ORSA J. Comput., 1(3). https://doi.org/10.1287/ijoc.1.3.190.

Grabusts, P., Musatovs, J. and Golenkov, V. (2019). The application of Simulated Annealing method for optimal route detection between objects. Procedia Comput. Sci. Elsevier B.V., 149. https://doi.org/10.1016/j.procs.2019.01.112.

Gracia, C., Diezma-Iglesias, B. and Barreiro, P. (2013). A hybrid genetic algorithm for route optimization in the bale collecting problem. Spanish Journal of Agricultural Research, 11(3). https://doi.10.5424/sjar/2013113-3635.

Han, Z., Wang, D., Liu, F., Zhao, Z. (2017). Multi-AGV path planning with double-path constraints by using an improved genetic algorithm. PLoS ONE, 12(7). https://doi.org/10.1371/journal.pone.0181747

He, P., Li, J., & Wang, X. (2018) Wheat harvest schedule model for agricultural machinery cooperatives considering fragmental farmlands. Comput. Electron. Agric., 145. https://doi.org/10.1016/j.compag.2017.12.042

He, Z., Bao, Y., Yu, Q., Lu, P., He, Y., Liu, Y. (2023). Dynamic path planning method for headland turning of unmanned agricultural vehicles. Computers and Electronics in Agriculture, 236. https://doi.org/10.1016/j.compag.2023.107699.

Hegazy, E.; Makhlouf, M.A.; El-Tawel, G.S. (2018). Dimensionality Reduction Using an Improved Whale Optimization Algorithm for Data Classification. Int. J. Mod. Educ. Comput. Sci., 10. https://doi.org/10.5815/ijmecs.2018.07.04

Hilal, Y. Y. et al. (2018). Development of genetic algorithm for optimization of yield models in oil palm production. Chilean journal of agricultural research, 78(2). https://doi.10.4067/S0718-58392018000200228.

Hu, Y., Liu, Y., Wang, Z., Wen, J., Li, J., & Lu, J. (2020). A two-stage dynamic capacity planning approach for agricultural machinery maintenance service with demand uncertainty. Biosyst. Eng., 190. https://doi.org/10.1016/j.biosystemseng.2019.12.005

Hussein, M.A., Antille, D.L., Kodur, S., Chen, G., Tullberg, J.N. (2021). Controlled traffic farming effects on productivity of grain sorghum, rainfall and fertiliser nitrogen use efficiency. Journal of Agriculture and Food Research, 3. https://doi.org/10.1016/j.jafr.2021.100111.

Ikeda, Y. and Inoue, M. (2016). An Evacuation Route Planning for Safety Route Guidance System after Natural Disaster Using Multi-objective Genetic Algorithm. Procedia Computer Science. The Author(s), 96. https://doi.10.1016/j.procs.2016.08.177.

Joo, H. and Lim, Y. (2018). Ant colony optimized routing strategy for electric vehicles. J. Adv. Transport. https://doi.org/10.1155/2018/5741982.

Kaur, S., Awasthi, L.K., Sangal, AlL., Dhiiman, G. (2020). Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Engineering Applications of Artificial Intelligence, 90. https://doi.org/10.1016/j.engappai.2020.103541.

Kellegöz, T., Toklu, B. and Wilson, J. (2008). Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. Applied Mathematics and Computation, 199(2). https://doi.org/10.1016/j.amc.2007.10.013.

Kennedy, J. a Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of ICNN95 - International Conference on Neural Networks, Perth, WA, Australia, 4. https://doi.org/10.1109/ICNN.1995.488968.

Khishe, M. et Mosavi, M.R. (2020). Chimp optimization algorithm. Expert Syst. Appl., 149. https://doi.org/10.1016/j.eswa.2020.113338

Koca, G. O., Dogan, S. and Yilmaz, H. (2018). A multi-objective route planning model based on genetic algorithm for cuboid surfaces. Automatika, 59(1). https://doi.10.1080/00051144.2018.1498205.

Kong, Q., Kuriyan, K., Shah, N., & Guo, M. (2019). Development of a responsive optimisation framework for decision-making in precision agriculture. Comput. Chem. Eng., 131. https://doi.org/10.1016/j.compchemeng.2019.106585

Kumhála F., Gutu D., Hůla J., Chyba J., Kovaříček P., Krouhlík M., Kvíz Z., Mašek J., Vlášková M. (2013). Technologie řízených přejezdů po pozemcích: uplatněná certifikovaná metodika. Česká zemědělská univerzita, Technická fakulta, Praha. ISBN 978-80-213-2425-1.

Kurnia, H. et al. (2018). Vehicle Routing Problem Using Genetic Algorithm with Multi Compartment on Vegetable Distribution. IOP Conference Series: Materials Science and Engineering, 325(1). https://doi.10.1088/1757-899X/325/1/012012.

Kwaśniewski, K. K. et Gosiewski, Z. (2018). Genetic algorithm for mobile robot route planning with obstacle avoidance. Acta Mechanica et Automatica, 12(2). https://doi.10.2478/ama-2018-0024.

Lamini, C., Benhlima, S. and Elbekri, A. (2018). Genetic algorithm based approach for autonomous mobile robot path planning. Procedia Computer Science. Elsevier B.V., 127. https://doi.10.1016/j.procs.2018.01.113.

Lee, H. Y., Shin, H. and Chae, J. (2018). Path planning for mobile agents using a genetic algorithm with a direction guided factor. Electronics (Switzerland), 7(10). https://doi.10.1016/S1067 2516(09)80071-9.

Leitold, D., Vathy-Fogarassy, A., & Abonyi, J. (2018). Network Distance-Based Simulated Annealing and Fuzzy Clustering for Sensor Placement Ensuring Observability and Minimal Relative Degree. Sensors, 18(9). https://doi.org/10.3390/s18093096

Li, X. et al. (2020). An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. J. Control Sci. Eng. https://doi.org/10.1155/2020/3857894.

Liu, L. et al. (2020). Harmony search method with global sharing factor based on natural number coding for vehicle routing problém. Information (Switzerland), 11(2). https://doi.org/10.3390/info11020086.

Mahaleh, M.B.B. et Mirroshandel, S.A. (2018). Harmony search path detection for vision based automated guided vehicle. Robot. Auton. Syst., 107. https://doi.org/10.1016/j.robot.2018.06.008

Marinello, F., Pezzuolo, A., Cillis, D., Chiumenti, A., & Sartori, L. (2017). Traffic effects on soil compaction and sugar beet (Beta vulgaris L.) taproot quality parameters. Spanish Journal of Agricultural Research, 15(1). https://doi.org/10.5424/sjar/2017151-8935

Mařík, V., Štěpánková, O., Lažanský, J. (2001). Umělá inteligence 3. Praha: Academia. ISBN 80-200-0472-6.

Mavrovouniotis, M., Ellinas, G., Polycarpou, M. (2018). Ant Colony Optimization for the Electric Vehicle Routing Problem. 8th IEEE Symposium Series on Computational Intelligence (IEEE SSCI). Bangalore, India, 2018. https://doi.org/10.1109/SSCI.2018.8628831.

Mileusnić, Z.I, Saljnikov, E., Radojević, R.L, Petrović, D.V. (2022). Soil compaction due to agricultural machinery impact. Journal of Terramechanics, 100. https://doi.org/10.1016/j.jterra.2021.12.002.

Mohammed, M. A. et al. (2017). Solving vehicle routing problem by using improved genetic algorithm for optimal solution. Journal of Computational Science. Elsevier B.V., 21. https://doi.10.1016/j.jocs.2017.04.003.

Mohiuddin, M. A., Khan, S. A. and Engelbrecht, A. P. (2014). Simulated evolution and Simulated Annealing algorithms for solving multi-objective open shortest path first weight setting problém. Appl. Intell., 41(2). https://doi.org/10.1007/s10489-014-0523-3.

Moon, I. et al. (2016). Evolutionary resource assignment for workload-based production scheduling. J. Intell. Manuf. Springer US, 27(2). https://doi.org/10.1007/s10845-014-0870-2.

Moriguchi, K. (2020) Acceleration and enhancement of reliability of Simulated Annealing for optimizing thinning schedule of a forest stand. Comput. Electron. Agric., 177. https://doi.org/10.1016/j.compag.2020.105691

Mukherjee, A., Misra, S., Sukrutha, A., & Raghuwanshi, N. S. (2020). Distributed aerial processing for IoT-based edge UAV swarms in smart farming. Computer Networks, 167. https://doi.org/10.1016/j.comnet.2019.107038

Mutar, M. L., Burhanuddin, M. A., Hameed, A. S., Yusof, N., & Mutashar, H. J. (2020). An efficient improvement of ant colony system algorithm for handling capacity vehicle routing problem. Int. J. Ind. Eng. Comput. https://doi.org/10.5267/j.ijiec.2020.4.006

Nazarahari, M., Khanmirza, E. and Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications. Elsevier Ltd, 115. https://doi.10.1016/j.eswa.2018.08.008.

Neungmatcha, W. et al. (2013). Adaptive genetic algorithm for solving sugarcane loading stations with multi-facility services problem. Computers and Electronics in Agriculture. Elsevier B.V., 98. https://doi.10.1016/j.compag.2013.07.016.

Ngoc, T. A., Hiramatsu, K. and Harada, M. (2014). Optimizing the rule curves of multi-use reservoir operation using a genetic algorithm with a penalty strategy. Paddy and Water Environment, 12(1). https://doi.10.1007/s10333-013-0366-2.

Notte, G. et al. (2016). Resource allocation in pastoral dairy production systems: Evaluating exact and genetic algorithms approaches. Agricultural Systems. Elsevier B.V., 148. https://doi.org/10.1016/j.agsy.2016.07.009.

Pamosoaji, A. K. and Setyohadi, D. B. Novel graph model for solving collision-free multiple-vehicle traveling salesman problem using ant colony optimization. Algorithms 2020, 13(6). https://doi.org/10.3390/A13060153.

Pohlheim, H. (2006). Evolutionary Algorithms: Overview, Methods and Operators. GEATbx - The Genetic and Evolutionary Algorithm Toolbox for Matlab, 8(December). https://doi.org/10.11909/j.issn.1671-5411.2018.04.004.

Prăvălie, R. (2021). Exploring the multiple land degradation pathways across the planet. Earth-Science Reviews, 103689. https://doi.org/10.1016/j.earscirev.2021.103689.

Pu, X. et al. (2020). 3D path planning for a robot based on improved ant colony algorithm. Evolutionary Intelligence. Springer Berlin Heidelberg. https://doi.org/10.1007/s12065-020-00397-6.

Qiongbing, Z. and Lixin, D. (2016). A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. Expert Systems with Applications. Elsevier Ltd, 60. https://doi.10.1016/j.eswa.2016.04.005.

Sales, L. de P. A., Pitombeira-Neto, A. R. and Prata, B. de A. (2018). A Genetic Algorithm Integrated with Monte Carlo Simulation for the Field Layout Design Problem. Oil & Gas Sciences and Technology – Revue d'IFP Energies nouvelles, 73. https://doi.10.2516/ogst/2018017.

Santos, L. C., Santos, F. N., Solteiro Pires, E. J., Valente, A., Costa, P., Magalhães, S. (2020). Path Planning for ground robots in agriculture: a short review. 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Ponta Delgada, Portugal, 2020. https://doi.10.1109/ICARSC49921.2020.9096177.

Sethanan, K., Theerakulpisut, S., Taechasook, P., Neungmatcha, W., Bureerat, S. (2013). Sugarcane Harvest Scheduling for Maximizing Total Sugar Yield. Adv. Sci. Lett., 19(10). https://doi.org/10.1166/asl.2013.5096

Seyyedhasani, H., Dvorak, J. S. and Roemmele, E. (2019). Routing algorithm selection for field coverage planning based on field shape and fleet size. Comput. Electron. Agric.. Elsevier, 156(March 2018). https://doi.org/10.1016/j.compag.2018.12.002.

Seyyedhasani, H., Dvorak, J.S. (2017). Using the Vehicle Routing Problem to reduce field completion times with multiple machines. Comput. Electron. Agr., 134. https://doi.org/10.1016/j.compag.2016.11.010

Shah, A.N., Tanveer, M., Shahzad, B. et al. (2017). Soil compaction effects on soil health and cropproductivity: an overview. Environ Sci Pollut Res, 24. https://doi.org/10.1007/s11356-017-8421-y

Shaheb, M.R., Venkatesh, R. & Shearer, S.A. (2021). A Review on the Effect of Soil Compaction and its Management for Sustainable Crop Production. J. Biosyst. Eng., 46. https://doi.org/10.1007/s42853-021-00117-7

Sivarajan, S., Maharlooei, M., Bajwa, S.G., Nowatzki, J. (2018). Impact of soil compaction due to wheel traffic on corn and soybean growth, development and yield. Soil and Tillage Research, 175. https://doi.org/10.1016/j.still.2017.09.001.

Soto, M. et al. (2017). Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problém. Comput. Ind. Eng., 107. https://doi.org/10.1016/j.cie.2017.03.022.

Sun, X. et al. (2018). Genetic Algorithm for Optimizing Routing Design and Fleet Allocation of Freeway Service Overlapping Patrol. Sustainability, 10(11). https://doi.org/10.3390/su10114120.

Tamirat, T.W, Pedersen, S.M., Farquharson, R.J., de Bruin, S., Forristal, P.D., Sørensen, C.G., Nuyttens, D., Pedersen, H.H., Thomsen, M.N. (2022). Controlled traffic farming and field traffic management: Perceptions of farmers

groups from Northern and Western European countries. Soil and Tillage Research, 217. https://doi.org/10.1016/j.still.2021.105288.

Tong, J. et al. (2017). Optimizing the path of seedling low-density transplanting by using greedy genetic algorithm. Computers and Electronics in Agriculture. Elsevier B.V., 142. https://doi.10.1016/j.compag.2017.09.017.

Utamima, A., Reiners, T., & Ansaripoor, A. H. (2019). Evolutionary Estimation of Distribution Algorithm for Agricultural Routing Planning in Field Logistics. Procedia Comput. Sci., 161. https://doi.org/10.1016/j.procs.2019.11.156

Utamina, A. et Djunaidi, A. (2021). Agricultural routing planning: A narrative review of literature. Procedia Computer Science, 197. https://doi.org/10.1016/j.procs.2021.12.190.

Vahdanjoo, M., Madsen, C. T. and Sørensen, C. G. (2020). Novel Route Planning System for Machinery Selection. Case: Slurry Application. AgriEngineering, 2(3). https://doi.org/10.3390/agriengineering2030028.

Valente, J., Del Cerro, J., Barrientos, A., Sanz, D. (2013). Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach. Comput. Electron. Agr., 99. https://doi.org/10.1016/j.compag.2013.09.008

Wieczorek, Ł. and Ignaciuk, P. (2018). Continuous Genetic Algorithms as Intelligent Assistance for Resource Distribution in Logistic Systems. Data, 3(4). https://doi.10.3390/data3040068.

Villar, J. R. et al. (2016). Genetic algorithm optimisation of heavy timber trusses with dowel joints according to Eurocode 5. Biosystems Engineering, 144. https://doi.org/10.1016/j.biosystemseng.2016.02.011.

Xing, L. et al. (2020). A novel tabu search algorithm for multi-AGV routing problem. Mathematics, 8(2). https://doi.org/10.3390/math8020279.

Yi, Y., Choi, K. and Lee, Y. (2016). Optimal Limited-stop Bus Routes Selection Using a Genetic Algorithm and Smart Card Data. Journal of Public Transportation, 19(4). https://doi.10.5038/2375-0901.19.4.11.

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam ukázek zdrojového kódu