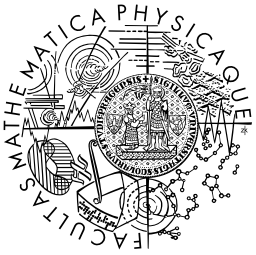


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Tomáš Karella

Evoluční algoritmy pro řízení heterogenních robotických swarmů

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Martin Pilát, Ph.D.

Studijní program: Informatika

Studijní obor: Programování a Softwarové Systémy

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Poděkování.

Název práce: Evoluční algoritmy pro řízení heterogenních robotických swarmů

Autor: Tomáš Karella

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Martin Pilát, Ph.D., katedra

Abstrakt: Abstrakt.

Klíčová slova: klíčová slova

Title: Evolutionary Algorithms for the Control of Heterogeneous Robotic Swarms

Author: Tomáš Karella

Department: Department of Theoretical Computer Science and Mathematical Logic at Faculty of Mathematics and Physics

Supervisor: Mgr. Martin Pilát, Ph.D., department

Abstract: Abstract.

Keywords: key words

Obsah

Úvod

Využití robotických hejn(robotic swarms) patří mezi rentabilní metody pro řešení složitějších úkolů. Zdá se, že velký počet jednoduchých robotů dokáže plně nahradit komplexnější jedince. Dostatečná velikost hejna umožní řešení úloh, které by jednotlivec z hejna provést nesvedl. Navíc přináší několik výhod, díky kvantitě jsou odolnější proti poškození či zničení, neboli zbytek robotů pokračuje v plnění cílů. Dále výroba jednodušších robotů vychází levněji než komplexní jedinců, což přináší vhodnou výhodu pro práci v nebezpečném prostředí. Hejno také může pokrývat vícero různých úkolů než specializovaný robot, který bude při plnění úkolů lišících se od typu úloh zamýšlených při konstrukci mnohem více nemotorný a nejspíše pomalejší. Hejno pokryje větší plochu při plnění úkolů.

Existuje mnoho aplikací robotických hejn, vetšinou se používají k úlohů týkajících se průzkumu a mapování prostředí, hledání nejkratších cest, nasazení robotů v nebezpečných místech (**?**). Jako příklad můžeme uvést asistenci záchraným složkám při požáru (**?**). Mnoho projektů zabývající se řízením robotického hejna se inspiruje přírodou, používá se analogie k chování mravenců a jiného hmyzu (**?**). Objevují se i harwarové implementace chování hejn, zmiňme projekty Swarm-bots (**?**), Colias (**?**)

Elementárnost senzorů i efektorů jednotlivých robotů vybízí k použití genetického programování, jelikož prostor řešení je velmi velký a plnění cílu lze vhodně ohodnotit. Dokonce na toto téma také vzniklo několik vědeckých prací (**?**) (**?**).

Cíl práce

Všechny zmíněné práce používají pro tvorbu řídicích programů genetické programování (GP) pracují s homogenními hejny. Cílem této práce je vyzkoušet využití GP na generování chování hejna heterogenních robotů, tedy skupiny robotů, kde se objevuje několik druhů jedinců a společně plní daný úkol. V rámci práce byl sestaven program pro simulaci různých scénářů a poz jejich úspěšnosti v rámci GP. Byli zvoleny 3 odlišné scénáře, kde se objevují 2-3 druhy robotů.

Struktura práce

Rozdělení práce je následující. První kapitola je věnována obecnému úvodu do tématiky genetického programování.

1. Evoluční algoritmy

1.1 Historie

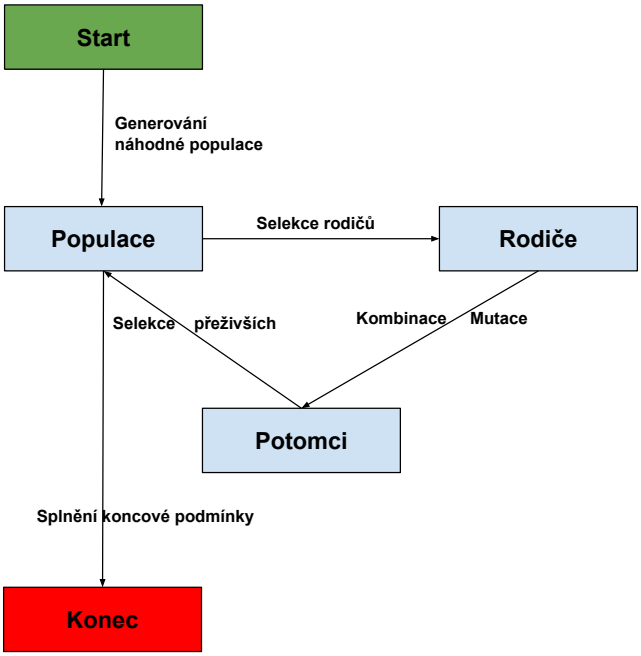
Začněme pohledem do historie Evolučních algoritmů na základě knih (?) a (?). Darwinova myšlenka evoluce lákala vědce už před průlomem počítačů, Turing vyslovil myšlenku *genetického a evolučního vyhledávání* už v roce 1948. V 50. a 60. letech nezávisle na sobě vznikají 4 hlavní teorie nesoucí podobnou myšlenku. Společným základem všech teorií byla evoluce populace kandidátů na řešení daného problému a jejich následná úprava způsoby hromadně nazývány jako genetické operátory, například mutace genů, přirozená selekce úspěšnějších řešení.

Rechenberg a Schwefell (1965, 1973) představuje *Evoluční strategie*, metoda optimalizující parametry v reálných číslech, jejich použití pro letadlová křídla. Fogel, Owens, Walsh zveřejňují *evolutionary programming*(evoluční programování), technika využívající k reprezentaci kandidátů konečný automat(s konečným počtem stavů), který je vyvíjen mutací přechodů mezi stavy a následnou selekcí. *Genetické algoritmy* vynalezl Holand v 60. letech a následně se svými studenty a kolegy z Michiganské Univerzity implementoval, oproti ES a EP nebylo hlavním cílem formovat algoritmus pro řešení konkrétních problémů, ale přenos obecného mechanismu evoluce jako metody aplikovatelné v informatickém světě. Princip GA spočívá v transformaci populace chromozonů(př. vektor 1 a 0) v novou populaci pomocí genetických operátorů křížení, mutací a inverze. V 1975 v knize *Adaptation in Natural and Artificial Systems* (?) definoval genetický algoritmus jako abstrakci biologické evoluce spolu s teoretickým základem jejich používání. Ovšem někteří vědci používají pojem GA i ve významech hodně vzdálených původní Holandově definici. K sjednocení jednotlivých přístupů přispěl v 90. letehc Koza, dále jsou všechny zahrnuty jako oblasti *Evolučních algoritmů*. Dnes existuje řada konferencí a odborných časopisů sdružující pracovníky zabývající se touto oblastí. Zmiňme ty větší z nich, co se týče konferencí: GECCO, PPSN, CEC, EVOSTAR, časopisy: Evolutionary Computation, IEEE Transactions on Evolutionary Computation, Genetic Programming and Evolvable Machines, Swarm and Evolutionary Computation

1.2 Co je evoluční algoritmus?

Ač existuje mnoho variant evolučních algoritmů, jak jsme zmínili v krátké pohledu do historie, spojuje je společná myšlenak populace jedinců uvnitř prostředí s omezenými podmínkami. Jedinci, jinak také nazývání kandidáti, soutěží o zdroje daného prostředí, tím je docíleno přírodní selekce(, přežijí jen Ti nejlepší). Pokud budeme mít k dispozici kvalitativní funkci, kterou se snažíme maximalizovat. Pak nebude problém vytvořit náhodné jedince z definičního oboru přesně této funkce. Náhodně vzniklé jedince můžeme ohodnotit, tímto způsobem dáme vzniku abstraktu pro měření fitness(, z anglického fit nejvíce vhodný). Z vzniklých a ohodnocených jedinců lze zvolit ty nejlepší pro tvorbu nové generace jedinců. Tvorba nové generace probíhá kombinováním zvolených rodičů a mutacemi jedinců. Jako kombinaci uvažujeme operátor, který je aplikován na 2-více zvolených kandidátů(, proto se jim také říká rodiče,) a tvořící 1-více nových jedinců(,

také nazývány děti). Mutace je aplikována pouze na 1 jedince a její výsledkem je také pouze 1 jedinec. Tyto dvě operace aplikované na rodičovskou generaci vedou k vytvoření nových kandidátů(potomků,offsprings). I tato nová generace je ohodnocena fitness a dále soutěží se starými jedinci na základě fitness(, občas také v závislosti na stáří kandidáta,) o místo v nové generaci. Popsaný proces je opakován dokud není nalezen kandidát s dostatečně velkou fitness nebo narazíme na výpoční limit(, bylo dosaženo požadovaného počtu opakovaní apod). Základy Evolučního systému pohání 2 základní hnací síly: variační operátory, selektivní operátory. Variační operátory zajišťují potřebnou různorodost v populaci a tím tvoří nové cesty k úspěšnému kandidátovi. Oproti tomu selektivní operátory zvyšují průměrnou kvalitu řešení v celé populaci. Kombinací těchto operátorů obecně vede ke zlepšování fitness hodnot v následující generaci. Celkem jednodušše lze vidět, zda evoluce optimalizuje či nikoliv, stačí nám k tomu pozorovat zda se fitness v popolaci blíží více a více k optimálním hodnotám vzhledem k postupu v čase. Mnoho komponent zapříčiňuje, že EA se řadí ke stochastickým metodům, selekce nevybírá nejlepší jedince deterministicky i jedinci s malou fitness mají šanci být rodiči následující generace. Během kombinování jsou části rodičů , kterou budou z zkombinovány, také zvoleny náhodně. Podobně u mutací, část která bude změněna je taktéž určena náhodně, nové rysy nahrazující staré jsou generovány taktéž náhodně.



Ze schéma na obrázku můžeme vyčíst, že EA patří mezi algoritmy ğenerate and test (vygeneruj a otestuj)ğ: Vyhodnocení fitness funkce poskytuje heuristický odhad kvality řešení, prohledávaní je řízen variací a selekcí. EA splňují charakteris-

tické rysy G&T algoritmů, zpracovávají zároveň celé kolekce kandidátů, většina EA míchá informace ze 2-více kandidátů a EA se řadí ke stochastickým metodám.

Různé dialekty evolučního programování, zmíněné v historickém okénku, splňují všechny tyto hlavní rysy a liší se pouze v technických detailech. Kandidáti jsou často reprezentování různě, liší se datové struktury pro jejich uchování i jejich zakódování. Typicky se jedná o řetězce nad konečnou abecedou v případě GA, vektory reálných čísel v ES, konečné automaty v EP a stromy v GP. Důvod těchto rozdílů je hlavně historický. Technicky však lze upřednostit jednu reprezentaci, pokud lépe odpovídá danému problému, tzn. kóduje kandidáta jednodušeji či přirozenější formou. Pro ilustraci zvolme řešení splnitelnosti(SAT) s n proměnnými, vcelku přirozeně sáhneme po použití řetězce bitů délky n a obsah i-tého bitu označuje, že i-tá proměnná má hodnotu (1)- pravda (0) - nepravda. Proto bychom použili jako vhodný EA Genetické Algortimy. Oproti tomu evoluce programu, který hraje šachy, by bylo vhodné použít derivační strom, kde by vrcholy odpovídali tahům na šachovnici. Přirozenější by tedy bylo použít GP.

Neopomeňme zmínit ještě dva důležité fakt. Za prvé kombinační a mutační operátory musí odpovídat dané reprezentaci, např. v GP kombinace pracuje se stromy(kombinuji podstromy..), zatímco v GA na řetězcích(prohazuji části řetězců). Za druhé oproti variacím musí fitness selekce záviset vždy pouze na fitness funkci, takže selekce pracuje nezávisle na reprezentaci.

1.3 Části evolučních algoritmů

Abychom vytvořili běhu schopný evoluční algoritmus, musíme specifikovat každou zmíněnou část a také inicializační funkci, která připraví první populaci. Pro konečný algoritmus ještě nesmíme opomenout a dodat koncovou podmínku.

- Reprezentace(definici individuí)
- Ohodnocující funce(Fitness funce)
- Populace
- Selekce rodičů
- Variační operátory(kombinace, mutace)
- Selekce prostředí

1.3.1 Reprezentace

Při tvorbě EA nejdříve musíme propojit prostor původního problému s prostorem řešení, kde se bude vlastní evoluce odehrávat. K docílení propojení je většinou potřeba zjednošit či vytvořit abstrakci nad aspekty reálného světa(prostor problému), abychom vytvořili vhodný prostor řešení, kde mohou být jednotlivá řešení ohodnocena. Neboli chceme docílit, aby možná řešení mohla být specifikována a uložena, tak aby s nimi mohl počítač pracovat. Objekty reprezentující možné řešení v kontextu původního problému jsou názývána **fenotyp**, zatímco kódování na straně EA prostoru **genotyp**. Reprezentace označuje mapování z fenotypů na genotypy, používá se ve významu funkci z fenotypu na genotyp(encode)

i genotypu na fenotyp(decode) a předpokládá se, že pro každý genotyp existuje nejvýše jeden fenotyp. Například pro optimalizační problém, kde množinou řešení jsou celá čísla. Celá čísla tvoří množinu fenotypu a můžeme se rozhodnout pro reprezentaci v binárních číslech. Tedy 18 by byl fenotyp a 10010 jeho genotyp. Prostor fenotypů a genotypů se zpravidla velmi liší a celý proces EA se odehrává pouze v genotypovém prostoru, vlastní řešení dostanema rozkódováním nejlepšího genotypu po ukončení EA. Jelikož nevíme, jak vlastní řešení vypadá, je nanejvýš vhodné umět reprezentovat všechny možná řešení, v G&T bychom řekli, že generátor je kompletní.

- V kontextu původního problému jsou následující výrazy ekvivaletní: fenotyp, kandidát(na řešení), jedinec, individuum (Množina všech možných řešení = fenotypový prostor)
- V kontextu EA: getotyp, chromozon, jedinec, individuum (Množina kde probíhá EA prohledávání = genotypový prostor)
- Části individuí jsou nazývány gen, locus, proměnná a dále se dělí na allely, či hodnoty

1.3.2 Populace

Populace je multimnožina genotypů, slouží jako jednotka evoluce. Populace utváří adaptaci a změny, zatímco vlastní jedinci se nijak nemění, jen vznikají nový a nahrazují předešlé. Pro danou reprezentaci je definice populace velmi jednoduchá charakterizuje ji pouze velikost. U některých specifických EA má populace další prostorové struktury, definované vzdáleností jedinců nebo relacemi sousedních jedinců. Což by se dalo připodobnit, reálným populacím, které se vyvíjejí v různých geografických prostředích. U většiny EA se velikost populace nemění, což vede k soutěživosti mezi jedinci(,zůstanou ti nejlepší). Na úrovni populací pracují právě selektivní operátory. Například zvolí nejlepší jedince aktuální populace jako rodiče následující, nahradí nejhorší jedince novými. Rozmanitost populace, vlastnost které chceme zpravidal docílit, je měřena jako počet různých řešení v multimnožině. Neexistuje však jediné hledisko podle, kterého lze tuto vlastnost měřit, většinou se používá počet rozdílných hodnot fitness, rozdílných fenotypů či genotypů a také například entropie(míra neuspořádanosti). Ovšem musíme mít na paměti, že jedna hodnota fitness v populaci neznamená, že populace obsahuje pouze jeden fenotyp, stejně tak jeden fenotyp nemusí odpovídat jednomu genotypu apod.

1.3.3 Selekce rodičů

Rodičovská selekce, někdy také partnerská selekce, vybírá lepší jedince jako rodiče pro příští generaci. Jedinec se stává rodičem, pokud byl zvolen k aplikaci variačních operátorů a tím dal vzniknout novým potomkům. Společně s selekcí přeživších je rodičovská selekce zoodpovědná za zvedání kvality v populacích. Rodičovská selekce je v EA většinou pravděpodobnostní metoda, která dává jedincům s větší kvalitou mnohem větší šanci býti vybrán než těm nízkou. Nicméně i jedincům s nízkou kvalitou je často přidělena malá nenulová pravděpodobnost

pro výběr, jinak by se celé prohledávání mohlo vydat slepou cestou a zaseknout se na lokální optimu.

1.3.4 Variační operátory

Hlavní úlohou variačních operátorů(mutace, rekombinaci) je vytváření nových individuí ze starých. Z pohledu Generate Test algoritmů spadají variační operátory do právě do Generate části. Obvykle je dělíme na dva typy podle jejich arity, jedná se o unární(Mutace) a n-arní(rekombinace) operátory.

Mutace

Ve většině případá myslíme unárním variačním operátorem mutace. Tento operátor je aplikován pouze na jeden genotyp a výsledkem je upravený potomek. Mutace řadíme mezi stochastické metody, výstup(potomek) totiž závisí na sérii náhodných rozhodnutí. Však mutace není vhodné pojmenování pro všechny variační operátory, například pokud se jedná o heuristiku závislou na daném problém, která se chová systematicky hledá slabé místo a následně se jej snaží vylepšit, nejedná se o mutaci v pravém slova smyslu. Obecně by mutace měla mutace způsobovat náhodné a nezaujaté změny. Unarní variační operátory hrají odlišnou roli v rozdílných EA opět díky historicky oddělenému vývoji. Zatímco v GP se nepoužívají vůbec, v GA má velmi důležitou roli a v EP se jedná o jediný variační operátor. Díky variačním operátorům aplikovaným v jednotlivých evolučních krocích dostává prohledávací prostor topologickou strukturu. Existují teorie formulící, že EA(s dostatečným časem) naleznou globální optimum daného problému opírající se právě o tuto topologii, spoléhají na vlastnost, že může vzniknout každý genotyp reprezentující možné řešení. Nejjednodušší cesta ke splnění těchto podmínek vede právě přes variační operátory. U mutací tohoto například dosáhneme, pokud povolíme, aby mohly skočit kamkoliv, tzn. každá allela může být zmutována na jakoukoli další s nenulovou pravděpodobností. Většina vědecká společnosti považuje tyto důkazy za nepříliš použitelné v praktickém využití a proto tuto vlastnost většina EA neimplementuje.

Rekombinace

Rekombinace, také nazývána křížení, je binární variační operátor. Jak název napovídá, spojuje informace ze 2 rodičů(genotypů) do 1 nebo 2 potomků. Stejně jako mutace patří rekombinace k stochastickým operátorům. Rozhodnutí, jaké části budou zkombinovány a jakým způsobem tak bude docíleno závisí na náhodě. Role rekombinace se znovu liší v rozličných EA, v GP se jedná o jediné variační operátory, v EP nejsou použity vůbec. Rekombinační operátory s vyšší aritou(, používající více než 2 rodiče,) jsou možné a jednoduché na implementaci, dokonce několik studií potvrdilo, že mají velmi pozitivní vliv na celou evoluci, ale nemají tak hojné zastoupení, nejspíše proto, že neexistuje biologický ekvivalent. Rekombinace funguje na jednoduchém principu, slučuje 2 individua a může vyprodukovat potomky, které kombinují jejich výhodné vlastnosti, tedy potomek je úspěšnější než jeho rodiče. Tento princip podporuje fakt, že po 1000-letí se aplikací rekombinace na rostliny a zemědělská zvířata mnokrát podařilo vytvořit nové

jedince s vyšším výnosem či jinými výhodnými vlastnostmy(odolnosti pro škůdcům,atd). Evoluční algoritmy vytváří množství potomků náhodnou rekombinací a doufáme, že zatímco malá část nové generace bude mít nežádoucí vlastnosti, většina se nezlepší ani si nepohorší a konečně další malá část předčí jejich rodiče. Na naší planetě se sexuálně(kombinací dvou jedinců) rozmnožují pouze vyšší organismy, což vzbuzuje dojem, že rekombinace je nejvyšší forma reprodukce. Neopomeňme, že variační operátory jsou závislé na reprezentaci(genotypu) jedinců. Např. pro genotypy bitových řetězců může použít bitovou inverzi jako vhodný mutační operátor ovšem pokud bude genotyp strom musíme zvolit nějaký jiný.

Selekce prostředí

Selekce prostředí, někdy také nazývána selekce přeživších, jak název napovídá má blízko k selekci rodičů, odlišuje jednotlivá individua na základě jejich fitness hodnoty. Oproti rodičovské selekci ji však používáme v jiné části evolučního cyklu, selekce prostředí spouštíme hned po vytvoření nových potomků. Jelikož velikost populace kandidátů bývá skoro vždy konstatní, tak je nutné rozhodnout které kandidáty zvolit do další generace. Toto rozhodnutí většinou závisí na fitnes kandidátů, také se však hledí na věk daného kandidáta(,generace v které vznikl). Na rozdíl od rodičovské selekce, která bývá stochastická, bývá selekce prostředí deterministickou metodou. Uveďme dvě nejběžnější metody, obě kladou největší důraz na fitness, první vybírá na nejlepší segment z množiny nových potomků i původní generace nezávisle, druhá dělá totéž jen z množiny nových potomků. Selekce prostředí je uváděna i pod názvem záměná strategie(replacement strategy), selekce prostředí(přeživších) se více používá pokud množina nových potomků je větší než velikost generace a záměna, když nových potomků je velmi málo.

Incializace populace

Inicializace populace zastává důležitý úkol vytvořit první generaci kandidátů. Zpravidla bývá ve většině EA její implementace velmi jednoduchá, první generace je vygenerována čistě náhodně. Principiálně by se zde dalo využít nějaké heuristiky k vytvoření vyšší fitness v první generaci, ovšem musela by zohledňovat řešený problém. Jestli tento postup stojí za výpočetní čas ne velmi záleží na konkrétní aplikaci EA. Ovšem existují obecná doporučení, která se touto otázkou zabývají.

Ukončovací podmínka

Rozlišme dvě varianty vhodné ukončovací podmínky. Pokud řešený problém má známou optimální hodnotu fitness, potom v ideální případě ukončovací podmínkou je řešní s touto fitness. Pokud jsme si vědomi, že náš model oproti modelovanému prostředí obsahuje nutná zjednoduššení nebo by se v něm mohly vyskytovat nežádoucí šumy, lze se spokojit s řešením, které dosáhlo optima fitness s přesností $\epsilon > 0$. Nicméně, EA díky své stochastičnosti většinou nemohou garantovat dosažení takovéto hodnoty fitness, takže ukončovací podmínka by nemusela být nikdy splněna a algoritmus by běžel věky. Kdybychom vyžadovali konečnost algoritmu, tak musíme rozšířit ukončovací podmínku. Zatímto účelem se nejběžněji používají následující rozšíření:

1. Byl překročen čas vyhrazený pro počítání na CPU
2. Celkový počet evaluací fitness přesáhl svůj předem daný limit
3. Zlepšení fitness v dané časovém bloku(, měřenému počtem generací, či evaluací) klesla pod přípustný práh
4. Rozmanitost v populaci nedosahuje předem určených hodnot

Prakticky ukončovací podmínka bývá disjukce: dosáhli jsme optima or podmínka X ze seznamu. Může se stát, že optimum známé není. Pak se používá pouze zmíněných podmínek nebo se smíříme s nekonečností algoritmu.

1.4 Differciální evoluce

Differrenciální evoluce(DE) se objevila v roce 1995, kdy Storn a Price vydali technický report TODO:citace popisující hlavní koncept skrývající se za pojmem DE N̈ew heuristic approach for minimizing possibly nonlinear and nondifferentiable continuous space functions.̈ Tento evoluční algoritmus popíši trochu více, protože je klíčovým algoritmem mého řešení.

Charakteristické vlastnosti

DE dostala své jméno hlavně díky změnám obvyklých reprodučkních operátorů v EA. Differciální mutace, jak jsou mutace v DE nazývány, z dané populace kandidátů vektorů v $\mathbb{R}^n$ vzniká nový mutant $\overline{x}'$ přičtením pertubačního vektoru(pertubation vector) k existujícímu kandidátovi.

$\overline{x}' = \overline{x} + \overline{p}$

Kde pertubační vektor $\overline{p}$ je vektor rozdílů dvou náhodně zvolených kandidátů z populace $\overline{y}a\overline{z}$.

$\overline{p} = F(\overline{y} - \overline{z})$

Škálovací faktor $F > 0$, $F \in \mathbb{R}$, který kontroluje míru mutace v populaci. Jako rekombinační operátor v DE slouží uniformní křížení, pravděpodobnost křížení je dáne $C_r \in [0,1]$, která definuje šanci jakékoli pozice v rodiči, že odpovídající allela potomka bude brána z 1. rodiče. DE také upravuje křížení, neboť 1 náhodná allela je brána vždy z 1. rodiče, aby nedocházelo k duplikaci 2. rodiče.

V hlavních implementacích DE reprezentují populace spíše listy, odpovídají lépe než množiny, umožňují referenci i-tého jedince podle pozice $i \in 1,..,\mu$ v listu. Pořadí kandidátů v této populaci $P =< \overline{x_1}...\overline{x_i}...\overline{x_\mu} >$ není závislé na hodnotě fitness.Evoluční cyklus začíná vytvořením populace vektorů mutantů $M =< \overline{v_1}...\overline{v_\mu} >$. Pro každého nového mutanta $\overline{v_i}$ jsou zvoleny 3 vektory z P, base vektora a dva definující pertubační vektor. Po vytvoření populace mutantů V, pokračujeme tvorbou populace $T =< \overline{u_1},....\overline{u_\mu} >$, kde $\overline{u_i}$ je výsledek aplikace křížení $\overline{u_i}$ je výsledek křížení $\overline{v_i}$ a $\overline{x_i}$(všimněme si, že je zaručené, že nezreplikujeme $\overline{x_i}$). Jako poslední aplikujeme selekci na každý pár $\overline{x_i}$ a $\overline{u_i}$ a do další generace vybereme $\overline{u_i}$ pokud $f(\overline{u_i}) \leq f(\overline{x_i})$ jinak $\overline{x_i}$.

DE algoritmus upravují 3 parametry, ?scalovací? faktor F, velikost populace μ (obvykle značen NP v DE literatuře) a pravděpodobnost křížení C_r. Na C_r lze také pohlížet jako na míru mutace, tzn. allela nebude odděděna od mutanta.

Vlastnosti Differciální Evoluce:

Reprezentace:	vektor $\mathbb{R}^n$
Rekombinace:	uniformní křížení
Mutace:	differeční mutace
Rodičovská selekce:	uniformní náhodné selekce 3 nezbytných vektorů
Selekce prostředí:	deterministická selekce elity (dítě vs rodič)

Varianty DE:

Během let, vzniklo a bylo publikováno mnoho variant DE. Jedna z modifikací zahrnuje možnost base vektoru, když vytváříme mutatnské populace M. Může být náhodně zvolen pro každé v_i, jak bylo řečeno, ale také lze využít jen nejlepšího vektoru a nechat změny na pertubačním vektoru. Další možnost otevírá použitím více pertubačních vektorů v mutačním operátoru. Což by vypadalo následovně:

$\overline{p} = F(\overline{y} - \overline{z} + \overline{y}' - \overline{z}')$, kde $\overline{y}, \overline{z}, \overline{y}', \overline{z}'$ jsou náhodně vybrány z původní populace.

Abychom odlišili různé varianty, používá se následující notace DE/a/b/c, kde **a** specifikuje base vektor(rand, best), **b** specifiuje počet differečních vektorů při vzniku pertubačního vektoru, **c** značí schéma křížení (bin=uniformní). Takže podle této notace by základní verze DE byla zapsána takto: DE/rand/1/bin.

2. Odkazy na literaturu

Odkazy na literaturu vytváříme nejlépe pomocí příkazů `\citet`, `\citep` atp. (viz LaTeXový balíček natbib) a následného použití BibTeXu. V matematickém textu obvykle odkazujeme stylem „Jméno autora/autorů (rok vydání)“, resp. „Jméno autora/autorů [číslo odkazu]“. V českém/slovenském textu je potřeba se navíc vypořádat s nutností skloňovat jméno autora, respektive přechylovat jméno autorky. Je potřeba mít na paměti, že standardní příkazy `\citet`, `\citep` produkují referenci se jménem autora/autorů v prvním pádě a jména autorek jsou nepřechýlena.

Pokud nepoužíváme bibTeX, řídíme se normou ISO 690 a zvyklostmi oboru.

Jména časopisů lze uvádět zkráceně, ale pouze v kodifikované podobě.

2.1 Několik ukázek

Mezi nejvíce citované statistické články patří práce Kaplana a Meiera a Coxe (**??**). **?** napsal článek o t-testu.

Prof. Anděl je autorem učebnice matematické statistiky (viz **?**). Teorii odhadu se věnuje práce **?**. V případě odkazů na specifickou informaci (definice, důkaz, ...) uvedenou v knize bývá užitečné uvést specificky číslo kapitoly, číslo věty atp. obsahující požadovanou informaci, např. viz **?**, Věta 4.22 nebo (viz **?**, Věta 4.22).

Mnoho článků je výsledkem spolupráce celé řady osob. Při odkazování v textu na článek se třemi autory obvykle při prvním výskytu uvedeme plný seznam: **?** představili koncept EM algoritmu. Respektive: Koncept EM algoritmu byl představen v práci Dempstera, Lairdové a Rubina (**?**). Při každém dalším výskytu již používáme zkrácenou verzi: **?** nabízejí též několik příkladů použití EM algoritmu. Respektive: Několik příkladů použití EM algoritmu lze nalézt též v práci Dempstera a kol. (**?**).

U článku s více než třemi autory odkazujeme vždy zkrácenou formou: První výsledky projektu ACCEPT jsou uvedeny v práci Genbergové a kol. (**?**). V textu *nenapíšeme*: První výsledky projektu ACCEPT jsou uvedeny v práci **?**.

3. Tabulky, obrázky, programy

Používání tabulek a grafů v odborném textu má některá společná pravidla a některá specifická. Tabulky a grafy neuvádíme přímo do textu, ale umístíme je buď na samostatné stránky nebo na vyhrazené místo v horní nebo dolní části běžných stránek. LaTeX se o umístění plovoucích grafů a tabulek postará automaticky.

Každý graf a tabulku očíslujeme a umístíme pod ně legendu. Legenda má popisovat obsah grafu či tabulky tak podrobně, aby jim čtenář rozuměl bez důkladného studování textu práce.

Na každou tabulku a graf musí být v textu odkaz pomocí jejich čísla. Na příslušném místě textu pak shrneme ty nejdůležitější závěry, které lze z tabulky či grafu učinit. Text by měl být čitelný a srozumitelný i bez prohlížení tabulek a grafů a tabulky a grafy by měly být srozumitelné i bez podrobné četby textu.

Na tabulky a grafy odkazujeme pokud možno nepřímo v průběhu běžného toku textu; místo *„Tabulka 3.1 ukazuje, že muži jsou v průměru o* 9,9 kg *těžší než ženy“* raději napíšeme *„Muži jsou o* 9,9 kg *těžší než ženy (viz Tabulka 3.1)“*.

3.1 Tabulky

U **tabulek** se doporučuje dodržovat následující pravidla:

- Vyhýbat se svislým linkám. Silnějšími vodorovnými linkami oddělit tabulku od okolního textu včetně legendy, slabšími vodorovnými linkami oddělovat záhlaví sloupců od těla tabulky a jednotlivé části tabulky mezi sebou. V LaTeXu tuto podobu tabulek implementuje balík `booktabs`. Chceme-li výrazněji oddělit některé sloupce od jiných, vložíme mezi ně větší mezeru.
- Neměnit typ, formát a význam obsahu políček v tomtéž sloupci (není dobré do téhož sloupce zapisovat tu průměr, onde procenta).
- Neopakovat tentýž obsah políček mnohokrát za sebou. Máme-li sloupec *Rozptyl*, který v prvních deseti řádcích obsahuje hodnotu 0,5 a v druhých deseti řádcích hodnotu 1,5, pak tento sloupec raději zrušíme a vyřešíme to jinak. Například můžeme tabulku rozdělit na dvě nebo do ní vložit popisné řádky, které informují o nějaké proměnné hodnotě opakující se v následujícím oddíle tabulky (např. *„Rozptyl* = 0,5 “ a níže *„Rozptyl* = 1,5 “).
- Čísla v tabulce zarovnávat na desetinnou čárku.

Efekt	**Odhad**	**Směrod. chyba**[a]	**P-hodnota**
Abs. člen	−10,01	1,01	—
Pohlaví (muž)	9,89	5,98	0,098
Výška (cm)	0,78	0,12	$< 0{,}001$

Pozn: [a] Směrodatná chyba odhadu metodou Monte Carlo.

Tabulka 3.1: Maximálně věrohodné odhady v modelu M.

- V tabulce je někdy potřebné používat zkratky, které se jinde nevyskytují. Tyto zkratky můžeme vysvětlit v legendě nebo v poznámkách pod tabulkou. Poznámky pod tabulkou můžeme využít i k podrobnějšímu vysvětlení významu některých sloupců nebo hodnot.

3.2 Obrázky

Několik rad týkajících se obrázků a grafů.

- Graf by měl být vytvořen ve velikosti, v níž bude použit v práci. Zmenšení příliš velkého grafu vede ke špatné čitelnosti popisků.
- Osy grafu musí být řádně popsány ve stejném jazyce, v jakém je psána práce (absenci diakritiky lze tolerovat). Kreslíme-li graf hmotnosti proti výšce, nenecháme na nich popisky `ht` a `wt`, ale osy popíšeme *Výška [cm]* a *Hmotnost [kg]*. Kreslíme-li graf funkce $h(x)$, popíšeme osy x a $h(x)$. Každá osa musí mít jasně určenou škálu.
- Chceme-li na dvourozměrném grafu vyznačit velké množství bodů, dáme pozor, aby se neslily do jednolité černé tmy. Je-li bodů mnoho, zmenšíme velikost symbolu, kterým je vykreslujeme, anebo vybereme jen malou část bodů, kterou do grafu zaneseme. Grafy, které obsahují tisíce bodů, dělají problémy hlavně v elektronických dokumentech, protože výrazně zvětšují velikost souborů.
- Budeme-li práci tisknout černobíle, vyhneme se používání barev. Čáry rozlišujeme typem (plná, tečkovaná, čerchovaná,...), plochy dostatečně rozdílnými intensitami šedé nebo šrafováním. Význam jednotlivých typů čar a ploch vysvětlíme buď v textové legendě ke grafu anebo v grafické legendě, která je přímo součástí obrázku.
- Vyhýbejte se bitmapovým obrázkům o nízkém rozlišení a zejména JPEGům (zuby a kompresní artefakty nevypadají na papíře pěkně). Lepší je vytvářet obrázky vektorově a vložit do textu jako PDF.

3.3 Programy

Algoritmy, výpisy programů a popis interakce s programy je vhodné odlišit od ostatního textu. Jednou z možností je použití LaTeXového balíčku `fancyvrb` (fancy verbatim), pomocí něhož je v souboru `makra.tex` nadefinováno prostředí `code`. Pomocí něho lze vytvořit např. následující ukázky.

```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

Menší písmo:

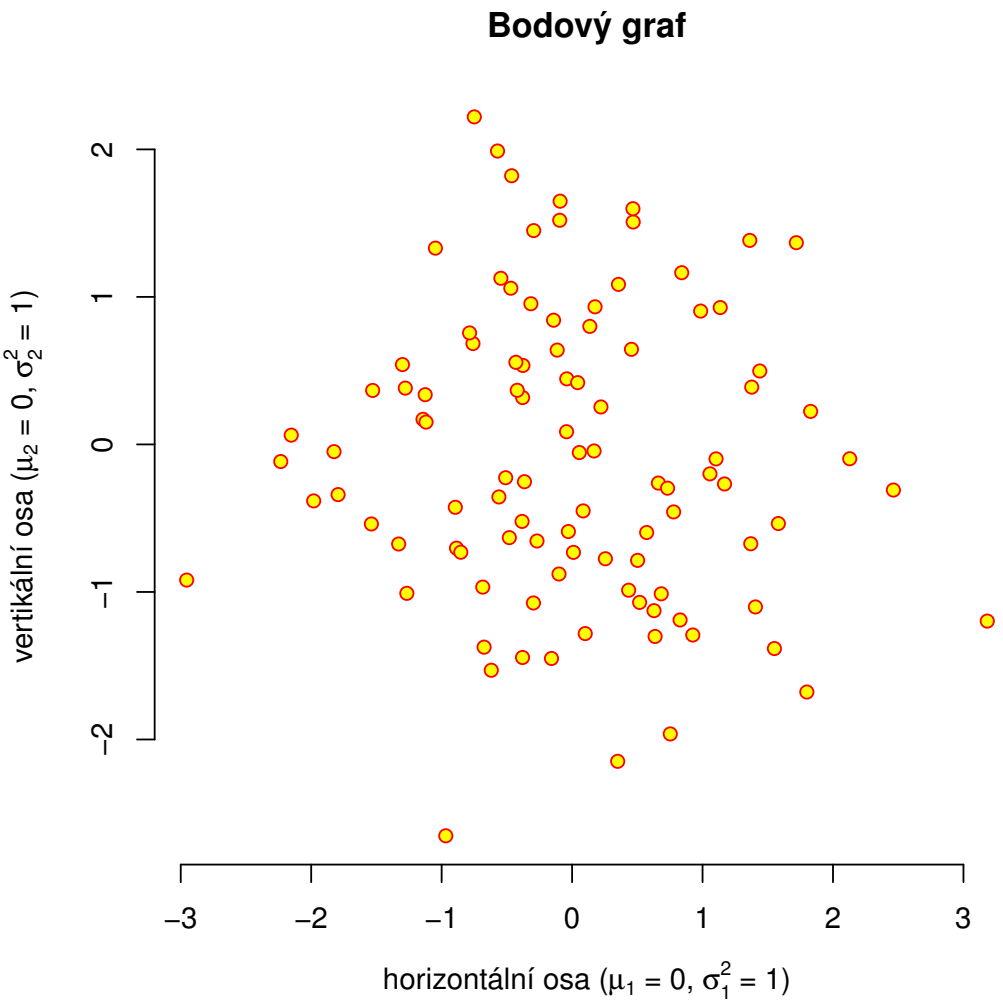
```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

Bez rámečku:

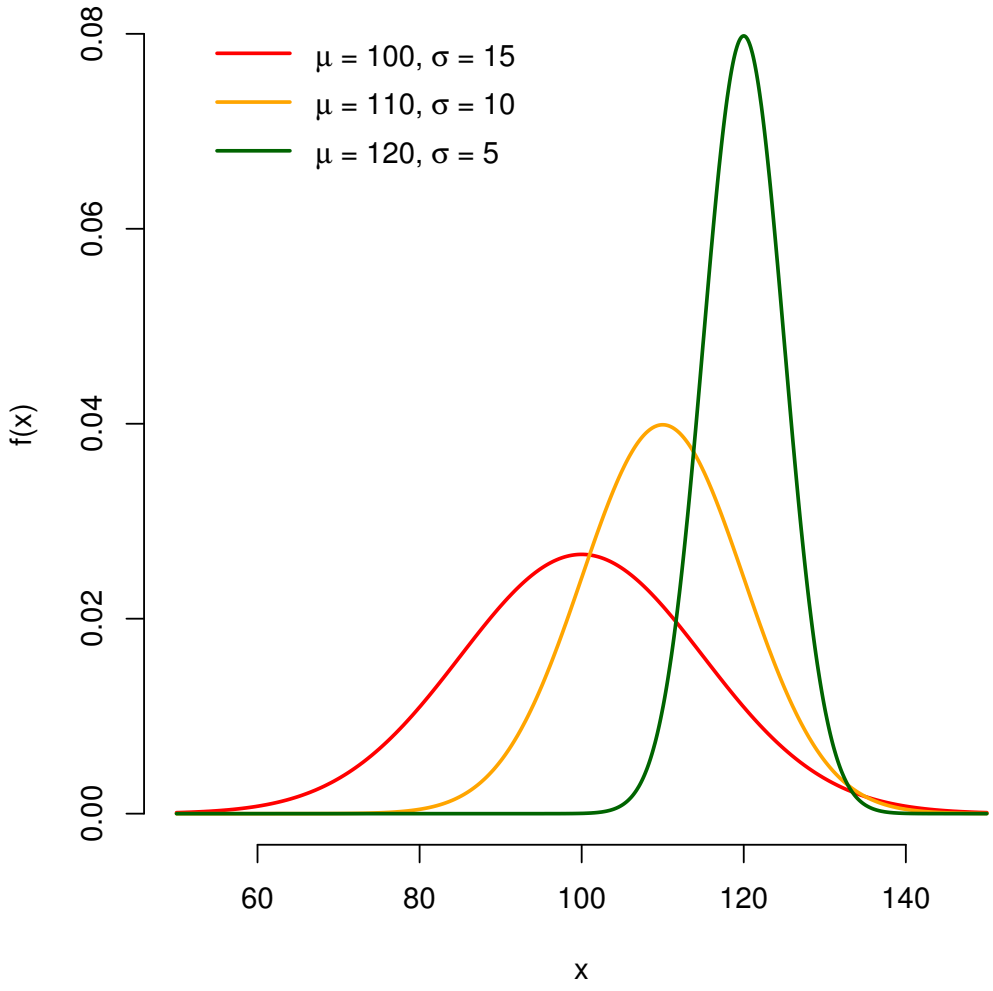
```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```

Užší rámeček:

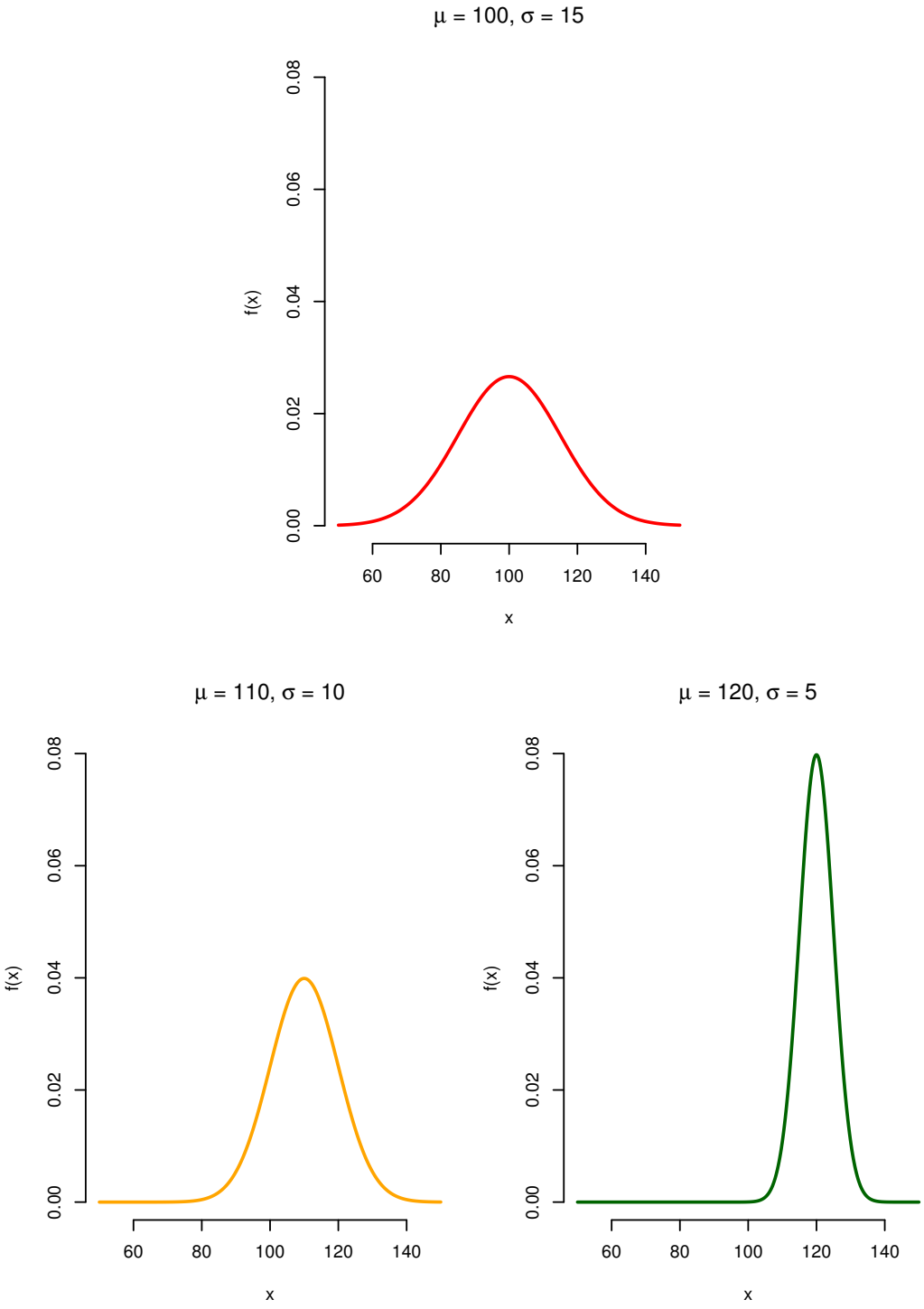
```
> mean(x)
[1] 158.90
> objekt$prumer
[1] 158.90
```



Obrázek 3.1: Náhodný výběr z rozdělení $\mathcal{N}_2(\mathbf{0}, I)$.



Obrázek 3.2: Hustoty několika normálních rozdělení.



Obrázek 3.3: Hustoty několika normálních rozdělení.

4. Formát PDF/A

Opatření rektora č. 23/2016 určuje, že elektronická podoba závěrečných prací musí být odevzdávána ve formátu PDF/A úrovně 1a nebo 2u. To jsou profily formátu PDF určující, jaké vlastnosti PDF je povoleno používat, aby byly dokumenty vhodné k dlouhodobé archivaci a dalšímu automatickému zpracování. Dále se budeme zabývat úrovní 2u, kterou sázíme TEXem.

Mezi nejdůležitější požadavky PDF/A-2u patří:

- Všechny fonty musí být zabudovány uvnitř dokumentu. Nejsou přípustné odkazy na externí fonty (ani na „systémové“, jako je Helvetica nebo Times).
- Fonty musí obsahovat tabulku ToUnicode, která definuje převod z kódování znaků použitého uvnitř fontu to Unicode. Díky tomu je možné z dokumentu spolehlivě extrahovat text.
- Dokument musí obsahovat metadata ve formátu XMP a je-li barevný, pak také formální specifikaci barevného prostoru.

Tato šablona používá balíček `pdfx`, který umí LaTeX nastavit tak, aby požadavky PDF/A splňoval. Metadata v XMP se generují automaticky podle informací v souboru `prace.xmpdata` (na vygenerovaný soubor se můžete podívat v `pdfa.xmpi`).

Validitu PDF/A můžete zkontrolovat pomocí nástroje VeraPDF, který je k dispozici na `http://verapdf.org/`.

Pokud soubor nebude validní, mezi obvyklé příčiny patří používání méně obvyklých fontů (které se vkládají pouze v bitmapové podobě a/nebo bez unicodových tabulek) a vkládání obrázků v PDF, které samy o sobě standard PDF/A nesplňují.

Je pravděpodobné, že se to týká obrázků vytvářených mnoha různými programy. V takovém případě se můžete pokusit obrázek do zkonvertovat do PDF/A pomocí GhostScriptu, například takto:

```
gs -q -dNOPAUSE -dBATCH
   -sDEVICE=pdfwrite -dPDFSETTINGS=/prepress
   -sOutputFile=vystup.pdf vstup.pdf
```

Jelikož PDF/A je v TEXovem světě (a nejen tam) novinkou, budeme rádi za vaše zkušenosti.

Závěr

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

Přílohy