Rozvrhování výroby surových plášťů pomocí evolučních výpočetních technik pro jeden stroj

Bc. Tomáš Silnica

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Silnica**
Osobní číslo:	**A15222**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Rozvrhování výroby surových plášťů pomocí evolučních výpočetních technik pro jeden stroj**
Téma anglicky:	**Scheduling Raw Tire Production Using Evolutionary Computational Techniques for One Machine**

Zásady pro vypracování:

1. **Proveďte analýzu problematiky rozvrhování výroby surových plášťů.**
2. **Seznamte se s optimalizačními evolučními algoritmy.**
3. **Navrhněte vhodně účelovou funkci a vhodný algoritmus.**
4. **Implementujte zvolené řešení.**
5. **Srovnejte navržené řešení se stávajícím.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ZELINKA I., OPLATKOVÁ Z., ŠEĎA M., OŠMERA P., VČELAŘ F. Evoluční výpočetní techniky: principy a aplikace. Praha: BEN – technická literatura, 2009. ISBN 978-80-7300-218-3.**
2. **GODFREY C. ONWUBOLU a B.V. BABU. New optimization techniques in engineering. Berlin: Springer, 2004. ISBN 9783642057670.**
3. **CONWAY, Richard Walter, William L. MAXWELL a L. W. MILLER. Theory of scheduling. Mineola, N.Y.: Dover, 2003. ISBN 0486428176.**
4. **TOMEK, Gustav a Věra VÁVROVÁ. Integrované řízení výroby: od operativního řízení výroby k dodavatelskému řetězci. Praha: Grada, 2014. Expert (Grada). ISBN 978-80-247-4486-5.**
5. **STORN, Rainer; PRICE, Kenneth. Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 1997, 11.4: 341-359.**
6. **HYNEK, Josef. Genetické algoritmy a genetické programování. Praha: Grada, 2008. ISBN 978-80-247-2695-3.**
7. **ZELINKA, Ivan, Václav SNÁŠEL a Ajith ABRAHAM. Handbook of optimization: from classical to modern approach. Berlin: Springer, c2013, xii, 1100 s. Intelligent systems reference library. ISBN 978-3-642-30503-0.**
8. **SIMON, Dan. Evolutionary optimization algorithms: biologically-inspired and population-based approaches to computer intelligence. Hoboken: Wiley, 2013, xxx, 742. ISBN 978-0-470-93741-9.**

Vedoucí diplomové práce: **doc. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 11.5.2017

.....
podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá problematikou rozvrhování výroby na jednom stroji pomocí evolučních výpočetních technik. Teoretická část poskytuje stručný popis základních typů úloh rozvrhování, seznámí čtenáře s výpočetní časovou složitostí a popisuje vybrané výpočetní evoluční techniky.

Praktická část práce obsahuje popis řešení problému rozvrhování na jednom stroji pomocí genetických algoritmů a diferenciální evoluce. Podrobně je popsán výpočet účelové funkce, která vyhodnocuje jednotlivé varianty řešení. Práce rovněž představuje řešení aplikace pro rozvrhování výroby surových plášťů ve výrobním podniku Continental Barum s.r.o.

Klíčová slova: rozvrhování, časová složitost, genetický algoritmus, diferenciální evoluce, účelová funkce

ABSTRACT

This diploma thesis deals with production scheduling on a single machine using evolutionary computational techniques. The theoretical part provides a brief description of the basic types of scheduling tasks and describes selected computational evolutionary techniques.

The practical part contains a description of the solution to the problem of scheduling on a single machine using genetic algorithms and differential evolution. It is described in detail the calculation the fitness function. The thesis also presents solutions for application scheduling green tire manufacturing plant in Continental Barum s.r.o.

Keywords: scheduling, time complexity, genetic algorithm, differential evolution, the fitness function

Na tomto místě bych rád poděkoval doc. Ing. Zuzaně Komínkové Oplatkové, Ph.D. za odborné vedení a konzultace, které mi pomohly při psaní této práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tato diplomová práce se věnuje problematice rozvrhování výroby surových plášťů pomocí evolučních výpočetních technik na jednom stroji. Cílem práce je vytvořit rozvrh výroby surových plášťů ve firmě Continental Barum s.r.o. Hlavní motivací je fakt, že v současné době je stále více kladen důraz na efektivní výrobu a zvyšování produktivity, proto optimalizace výrobního toku a vytvoření optimálního rozvrhu výroby jsou klíčové otázky pro udržení konkurenceschopnosti ve všech výrobních podnicích.

Teoretická část je věnována možnostem plánování výroby spolu s možnými kritérii pro rozvrhování, popisem výrobních systémů, určení časové náročnosti výpočtů při rozvrhování a také popisem vybraných evolučních technik, které lze použít při rozvrhování na jednom stroji.

V praktické části je realizován výpočet operativního rozvrhu výroby surových plášťů na jednom stroji. Požadavek vznikl u výrobce automobilových plášťů Continental Barum s.r.o, Podnik vyrábí sortiment výrobků, jenž je v současné době velmi rozmanitý, a tím jsou na výrobní provozy kladeny mnohem větší požadavky, než tomu bývalo dříve. Je nutné vyrábět velké množství výrobků v krátkém časovém období. S tím souvisí i složitější řízení výroby, a v určitém okamžiku se pro udržení kvality a efektivity stává nezbytností využití informačních technologií. Vzhledem k různorodosti a množství výrobků bylo k vytvoření rozvrhu výroby surových plášťů zvoleno řešení pomocí evolučních výpočetních technik, které řeší tento složitý kombinatorický problém ve velmi krátkém čase a při vyhovujícím řešením. Takto realizované řešení, pomocí nějž se vytváří rozvrh výroby, je aplikováno na 39 výrobních modulech. Výpočet je prováděn za pomocí genetických algoritmů nebo s využitím diferenciální evoluce. V této práci jsou obě tyto metody řešení problému vzájemně porovnány a vyhodnoceny. V závěru praktické části je shrnuto porovnání navrženého a stávajícího řešení.

I. TEORETICKÁ ČÁST

1 ROZVRHOVÁNÍ VÝROBY

Při rozvrhování výroby je snahou optimální přiřazení strojního zařízení dané množině úloh v konkrétním čase tak, aby bylo co nejvíce toto zařízení využito, a zároveň, aby byly splněny aktuální požadavky. Úkolem je při omezeném množství zdrojů maximalizovat zisk za daných omezení. Úlohy rozvrhování lze formulovat jako úlohy celočíselného programování a řešit pomocí optimalizačních metod [10].

Základní pojmy pro rozvrhování mohou být definovány jako [5]:

- operace O_i – základní technologický úkon,
- úloha J_i – posloupnost požadovaných operací,
- stroj M_i – strojní zařízení, které vykonává jednotlivé operace.

Statické parametry úlohy:

- doba trvání p_j – potřebný čas provádění úlohy,
- termín dostupnosti r_i – čas, kdy lze nejdříve úlohu J_i provádět,
- termín dokončení d_i – čas, kdy má být úloha dokončena,
- váha w_i – přiřazení důležitosti úlohy J_i

Dynamické parametry úlohy:

- čas zahájení úlohy S_i,
- čas ukončení úlohy C_j,
- čas pobytu úlohy v systému F_j,
- časová odchylka L_i od plánovaného ukončení úlohy J_i. Platí, že $L_i = C_i - d_i$,
- zpoždění úlohy T_i,
- penalizace zpoždění U_i. Když $U_i < Ci$ potom $U_i = 0$, jinak $U_i = 1$.

Prioritní pravidla rozvrhování

Rozvrhování podle prioritních pravidel pomáhá určit, kterou aktuálně dostupnou úlohu zpracovávat na stroji přednostně. Tyto pravidla vyplývají z obecných požadavků na výrobu nebo podle technologického postupu, a jsou nejčastěji určeny pomocí vah w_i.

Některé vybrané prioritní pravidla [2]:

- Prioritní – úloha s větší prioritou má přednost před úlohou s menší prioritou,
- FIFO systém – zařazení úloh v takovém pořadí, v jakém do systému vstoupily ostatní úlohy,
- LIFO systém – poslední úloha vstupuje do výroby jako první,
- nejdříve možný termín dokončení - úloha je zařazena před všechny úlohy, které mají pozdější termín dokončení,
- nejkratší výrobní čas – zařazení úlohy před úlohy s delším výrobním časem,
- nejmenší rozdíl mezi termínem dodání a zbývajícím časem výroby – zařazení úlohy před úlohy s větším rozdílem mezi termínem dodání a zbývajícím čase výroby.

1.1 Rozvrh a jeho kritéria

Rozvrh *p* pro konkrétní strojní zařízení je dán rozmístěním jednotlivých úloh do konkrétního času. Cílem je vytvořit optimální rozvrh, který nejlépe vyhovuje všem požadovaným optimalizačním kritériím.

Úplný rozvrh je uzpůsoben kritériu, které představuje podmínku, že v rozvrhu jsou obsaženy všechny úlohy pro daný čas. Následná tvorba přesného rozvrhu vyžaduje značné znalosti o jednotlivých úlohách a přidělených zdrojích [5].

Částečný rozvrh je vytvořen pouze pro vybrané úlohy. Není zde nutností zařadit do rozvrhu všechny úlohy. Úlohy jsou zařazeny do rozvrhu nejčastěji podle prioritních pravidel [5].

Konzistentní rozvrh splňuje všechna omezení, která jsou na stroj kladena. Za omezení můžeme například považovat to, že na jednom stroji neběží více úloh, než je jeho kapacita.

Optimální rozvrh je takový, kdy umístění úloh na stroj je optimální, a to vzhledem k zadanému optimalizačnímu kritériu. Příkladem požadovaného optimalizačního kritéria může být, např. čas dokončení poslední úlohy má být minimální [5].

Operativní rozvrh obsahuje vybrané úlohy, které jsou v dané době dostupné a požadované. Při vytváření rozvrhu je třeba dbát na dynamiku prostředí rozvrhu a případné poruchy strojů, které lze řešit vhodným způsobem nahrazení na ostatních strojích v rozumně krátké době [5].

Kritéria rozvrhu

Kvalita rozvrhu se určuje podle mnoha parametrů. Jedním z nejčastěji používaných minimalizačních kritérií je celková doba zpracování všech úloh [4].

Vybrané kritéria rozvrhu [4]:

Úkolem je najít takový rozvrh *p*, který minimalizuje čas dokončení poslední zpracovávané úlohy (1):

$$C_{max}(p) = \max_{1 \le j \le n}(C_{Last\ i}) \tag{1}$$

Minimalizace ztrát spojených s nedodržením termínů je možné vyjádřit například jako největší časová odchylka (2):

$$L_{max}(p) = \max_{1 \le i \le n}(L_i) \tag{2}$$

nebo také jako největší zpoždění úlohy (3):

$$T_{max}(p) = \max_{1 \le i \le n}(T_i) \tag{3}$$

Jestliže je uplatňována rozdílná důležitost úloh, je nutné aplikovat váhy w_i na uvedená minimalizační kritéria (4) - (6):

$$C^w_{max}(p) = \max_{1 \le i \le n}(w_i\, C_{Last\ i}) \tag{4}$$

$$L^w_{max}(p) = \max_{1 \le i \le n}(w_i\, L_i) \tag{5}$$

$$T^w_{max}(p) = \max_{1 \le i \le n}(w_i\, T_i) \tag{6}$$

Dalšími minimalizačními kritérii jsou sumy časů dokončení (7):

$$C_\Sigma(p) = \sum_{i=1}^{n}(C_{Last\ i}) \tag{7}$$

Minimalizace pomocí sumy časových odchylek od výrobního plánu (8):

$$L_\Sigma(p) = \sum_{i=1}^{n}(L_i) \tag{8}$$

Minimalizační kritérium pro sumy zpoždění všech úloh (9):

$$T_{\Sigma}(p) = \sum_{i=1}^{n} (T_i) \tag{9}$$

Minimalizace ztráty pomocí součtu časů pobytu všech úloh v systému (10):

$$F_{\Sigma}(p) = \sum_{i=1}^{n} (F_i) \tag{10}$$

Kritérium pro minimalizaci zpoždění úloh (11):

$$U_{\Sigma}(p) = \sum_{i=1}^{n} (U_i) \tag{11}$$

Jestliže je uplatňována rozdílná důležitost úloh, je nutné aplikovat váhy w_i na uvedená sumační minimalizační kritéria (12) - (16):

$$C_{\Sigma}^{w}(p) = \sum_{i=1}^{n} (w_i C_{Last\ i}) \tag{12}$$

$$L_{\Sigma}^{w}(p) = \sum_{i=1}^{n} (w_i L_i) \tag{13}$$

$$T_{\Sigma}^{w}(p) = \sum_{i=1}^{n} (w_i T_i) \tag{14}$$

$$F_{\Sigma}^{w}(p) = \sum_{i=1}^{n} (w_i F_i) \tag{15}$$

$$U_{\Sigma}^{w}(p) = \sum_{i=1}^{n} (w_i U_i) \tag{16}$$

1.2 Rozvrhování na jednom stroji

K základnímu modelu rozvrhování výroby patří model s jedním strojem, který dokáže zpracovávat vždy jen jednu úlohu. Přičemž každá úloha má stanoven termín dostupnosti r_i a termín dokončení d_i. Cílem je minimalizovat maximální překročení termínu dokončení $L_{i(max)}$. Obecně se jedná o NP-těžký problém [5].

1.3 Rozvrhování a paralelní stroje

Při zpracování úloh může být k dispozici více alternativních strojů, které mohou být zcela identické, nebo je lze rozdělit do kategorií podle jejich rozdílných parametrů.

Identické paralelní stroje představují všechny stroje, které mají stejnou dobu zpracování. Daná úloha může být prováděna na libovolném *m* stroji [5].

Paralelní stroje s různou rychlostí jsou stroje, kde doba zpracování úlohy je přímo úměrná rychlosti daného stroje. Pro výpočet doby pro paralelní stroj se násobí doba zpracování koeficientem rychlosti stroje, přičemž poměr mezi jednotlivými stroji bývá zachován [5].

Nezávislé paralelní stroje s různou rychlostí mají libovolnou dobu zpracování pro každý stroj. Není zde dodržen poměr rychlostí mezi jednotlivými stroji [5].

1.4 Multi-operační plánování

Často bývá úloha rozdělena na menší operace a poté prováděna postupně na několika strojích. Stroje jsou pro jednotlivé operace jednoznačně rozděleny [6].

Job-shop problém je určen množinou *m* strojů, *n* úloh. Pořadí provádění operací pro každou úlohu je předem určeno, přičemž čas stroj i čas výroby je pevně daný. Úloha je obvykle přidělena na konkrétní stroj pouze jedenkrát, ale je i možnost daný stroj využít i vícekrát. Job-shop problém je zařazován mezi NP-úplné problémy [6].

Flow-shop je multi-operační problém s *m* stroji v sérii, kdy každá úloha musí být prováděna na všech strojích [6].

Open-shop je multi-operační problém s *m* stroji. Není dáno technologickým postupem pořadí operací, proto se musí určit, v jakém pořadí budou úlohy prováděny. Doba zpracování úlohy na některých strojích může být i nulová [6].

2 VÝROBNÍ SYSTÉMY

Při plánování i samotném rozvrhování výroby ve výrobních podnicích je stále více kladeno na produktivitu práce. Investice do nového strojního zařízení, které mohou pomoci tuto produktivitu zvýšit, dosahují obvykle velmi vysokých finančních částek, a proto se hledají cesty k zefektivnění výroby na stávajícím zařízení. Toho lze poměrně rychle a levně dosáhnout pomocí zlepšení organizace a řízení výroby. K těmto změnám vede několik používaných metod, které nahlížejí na problematiku řízení výroby podle pevně nastavených pravidel.

2.1 MRP (Material Requirements Planning)

MRP neboli plánování materiálových potřeb, je způsob řízení založený na plánování materiálových potřeb, bez ohledu na výrobní zdroje a jejich omezení. Princip metody tkví v hledání rovnováhy mezi potřebou a zásobou. Tato metoda ale nebere v potaz aktuální průběh výroby, čímž lehce může nastat, že vznikne např. neplánované navýšení skladových zásob [16].

2.2 MRP II (Manufacturing Resource Planning)

MRP II je následovník MRP metody plánování a zahrnuje veškeré zdroje spojené s výrobou. Zahrnuje funkce plánování denního množství, kontrolní systémy a určování kritických částí výroby. Systém využívá principu tlaku (push), kdy se při maximálním využití kapacit vyrábí pouze to, co je naplánováno [16].

2.3 JIT (Just-in-time)

Metoda, která zajišťuje tok jednotlivých materiálů přesně v ten moment, kdy jsou potřeba. Tímto se minimalizují náklady na stav zásob, dochází ke zlepšení produktivity a snížení nadvýroby. JIT metoda klade velmi vysoké nároky na naprosto přesnou koordinaci všech souvisejících procesů a toků [16].

2.3.1 KANBAN

Systém dílenského řízení, jehož název pochází z japonského slova kan (karta) a ban (signál). Používá se princip tahu (pull), kdy se vytváří dvojice dodávající-odběratel, kteří využívají přesně stanovené samořídicí regulační okruhy. Systém využívá pro řízení toku materiálu kanban karty, které jsou komunikační prostředek a nositel informací, a zároveň

jsou pro dodávajícího impulsem k zahájení výroby jedné jednotky výrobku. Principem metody Kanban je vyrábět jen tolik, kolik potřebuje zákazník [2].

2.4 TOC (Theory of Constraints)

Základní myšlenkou této manažerské filozofie je tvrzení, že každý systém v sobě skrývá minimálně jedno úzké místo, neboli omezení. Tento přístup poskytuje metodiku, jak určité omezení nalézt a účinně je využívat. K tomu slouží základní kroky neustálého zlepšování pomocí Teorie omezení [15]:

- vyhledat omezení systému,
- zjistit, jak omezení maximálně využít,
- podřídit vše ostatní chodu omezení,
- navýšit kapacitu omezení,
- vrátit se k vyhledávání omezení systému.

2.4.1 DBR (Drum Buffer Rope)

DBR je založeno na regulaci vstupu výrobních úloh do výrobního systému podle průběhu činností na úzkých místech. Úzké místo udává tempo celému výrobnímu systému. Této funkci úzkého místa se tedy říká DRUM (buben). Úzké místo musí být chráněno vhodně dimenzovaným nárazníkem, čemuž se říká BUFFER (zásobník). Prostřednictvím „lana“ (ROPE) je úzké místo provázané se vstupem materiálu do výrobního systému [15].

2.5 APS (Advanced Planning & Scheduling)

Systém pokročilého plánování a rozvrhování v prostředí s omezenou kapacitou. Systém nejdříve určí existující kapacity zdrojů, teprve poté vytváří rozvrh úloh pro výrobní zdroje. Důraz je kladen na nepřekročení daných omezení. Úlohy jsou naplánovány do přesných časů a na jednotlivé výrobní zdroje. V systému APS se používá pokročilých matematických metod, optimalizačních procesů nebo využití simulace plánování, které jsou prováděny za pomocí výpočetní techniky [16].

3 ČASOVÁ SLOŽITOST

Pro řešení optimalizačního problému pomocí výpočetní techniky byly stanoveny postupy, které posuzují efektivnost a časovou náročnost použitých algoritmů.

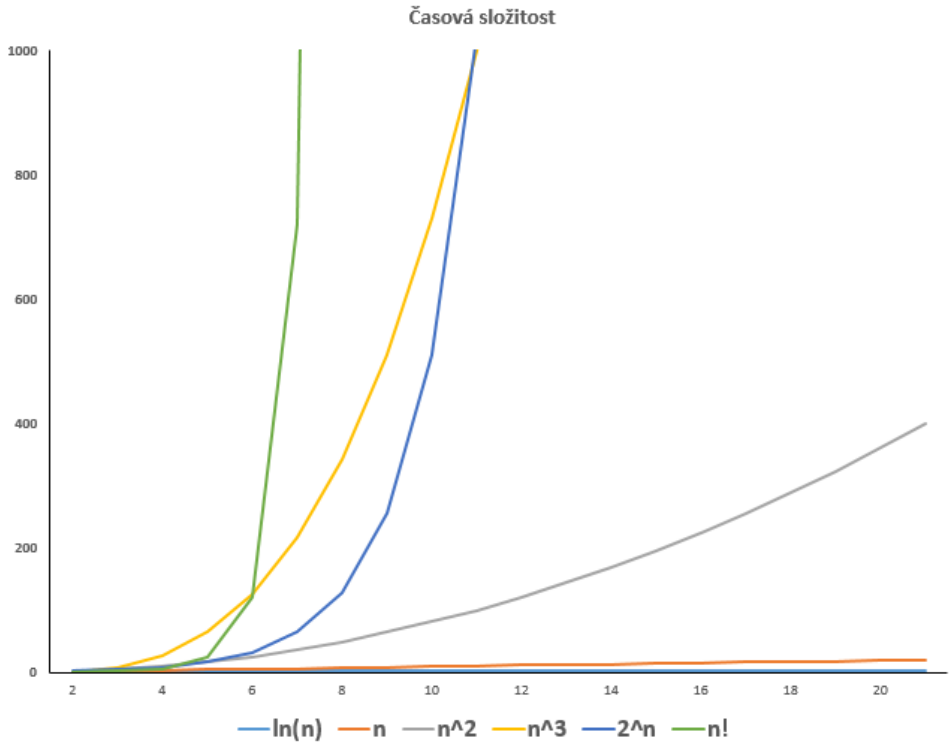
Časová složitost algoritmu vyjadřuje závislost času potřebného pro provedení výpočtu na rozsahu vstupních dat. Čas se v této souvislosti chápe jako počet provedených operací, a je uváděn jako bezrozměrná jednotka [7].

Obecně se rozlišuje časová složitost v nejlepším, průměrném a nejhorším případě. Vzhledem k tomu, že složitost algoritmů většinou neodpovídá pouze jedné třídě, používá se nejčastěji výpočet pro nejhorší možný případ [7].

Algoritmy dělíme podle časové složitosti na algoritmy s polynomiální a nepolynomiální časovou složitostí. Časovou složitost můžeme zapsat pomocí asymptotické notace, která vyjadřuje odhad rychlosti růstu funkce, přičemž jsou zanedbány konkrétní konstanty a méně významné členy. Příklad asymptotické notace je např. $f(n) = O(g(n))$, kde f a g jsou funkce definované na množině přirozených čísel a n je počet operací. Polynomiální říkáme těm, jejichž časová složitost je omezena polynomem, tj. $O(n^k)$, kde k je konstanta. Naopak nepolynomiální jsou ty, pro něž žádné takové k neexistuje [8].

Přehled nejčastěji se vyskytujících časových složitostí algoritmů seřazených od nejrychlejších po časově nejsložitější [8]:

- ***O(1)*** - konstantní (příklad: je číslo sudé?),
- ***O(log n)*** - logaritmická (binární vyhledávání),
- ***O(n)*** - lineární (hledání maxima z n čísel),
- ***O(n.log n)*** - lineárně-logaritmická (třídění pomocí porovnávání),
- ***O(n^2)*** – kvadratická (BubbleSort),
- ***O(n^3)*** - kubická (násobení matic podle definice),
- ***O(2^n)*** - exponenciální (nalezení všech posloupností délky n složených z 0 a 1),
- ***O(n!)*** - faktoriálová, n! = 1.2.3…. n (nalezení všech permutací n prvků) [17].



Obr. č. 1 Graf časových složitostí algoritmů.

3.1 Třídy složitosti

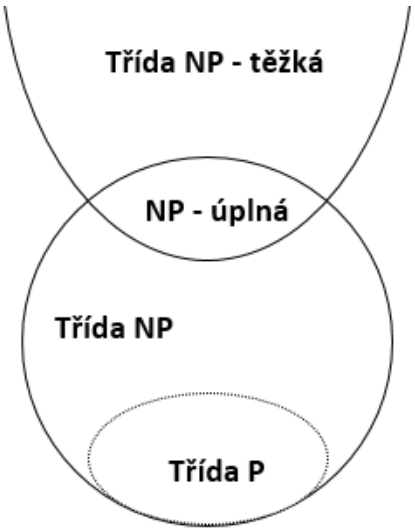
Každý algoritmus lze, na základě podobných rysů, rozdělit do tříd složitosti. Ke klasifikaci algoritmů se obvykle používá tzv. asymtotická složitost, což je rozdělení algoritmů do tříd složitostí, u kterých platí, že od určité velikosti dat je algoritmus dané třídy vždy pomalejší než algoritmus třídy předchozí. Snahou je zařadit problém do co nejnižší třídy složitosti.

Třída P je třída problémů, kterou lze poměrně efektivně vyřešit v polynomiálním čase, a jsou řešitelné pomocí deterministického Turingova stroje [9].

Třída NP (nedeterministicky polynomiální) je třída problémů, kterou lze řešit v omezeném polynomiálním čase na nedeterministickém Turingově stroji. V této třídě složitosti lze ověřit výsledek v polynomiálním čase [9].

Třída NP-úplná (NPC, NP-complete) je, pokud tato třída je součástí třídy NP a zároveň platí, že se na ní polynomiálně redukuje každá úloha z třídy NP. NP-úplné úlohy jsou ty „nejobtížnější“ ze všech NP úloh [9].

Třída NP-obtížná (NP-hard, NP-těžká) je, pokud se na ní redukuje NP-úplná úloha, ale zároveň nevíme, jestli je v NP. Taková úloha je minimálně stejně těžká jako všechny NP-úplné úlohy [9].



Obr. č. 2 Schéma diagramu tříd P, NP, NP-úplná a NP-těžká [17].

3.2 Řešení NP problémů

Třída NP problémů je kategorie problémů, pro které doposud nebylo nalezeno polynomiální řešení, nebo se předpokládá, a to s ohledem na jejich složitost, že pro jejich výpočet nebude nikdy nalezeno polynomiální řešení [10].

Pokud tedy pro daný problém neexistuje efektivní algoritmus, máme následující možnosti [11] :

- **Exponenciální algoritmy** – použitelné jen na malé instance. Algoritmus zkoumá všechny možnosti řešení,
- **Aproximační řešení** – použitelné jen pro optimalizační problémy, kdy algoritmus najde řešení, které je o něco horší než optimum,
- **Pravděpodobnostní algoritmy** – algoritmus najde rychle řešení, ale řešení je s určitou pravděpodobností nesprávné.
- **Speciální případy** - jen na konkrétní případy, neřeší problém v plné obecnosti.
- **Heuristika** - postup, který rychle najde přibližné řešení, ovšem není zaručeno, že toto řešení je nejlepší možné.
- **Metaheuristika** jsou heuristické postupy, které umožňující, za jistých podmínek, opustit lokální extrém a přejít do jiných částí množiny přípustných řešení, kde lze nalézt lepší hodnotu účelové funkce.

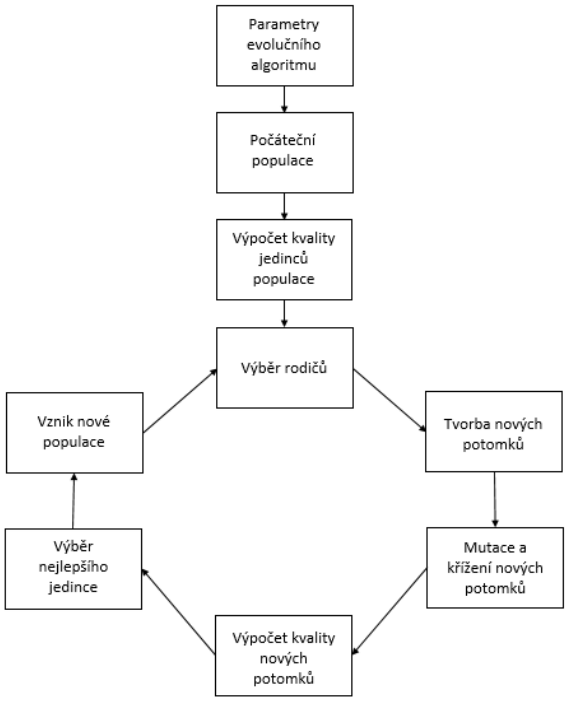
4 EVOLUČNÍ VÝPOČETNÍ TECHNIKY

Evoluční výpočetní techniky (EVT) jsou numerické algoritmy založené na Darwinově teorii evoluce, která je založena na třech základních principech, které představují přirozený výběr, náhodné ovlivnění jedince a reprodukční proces. Jednotlivé druhy se přirozeně vyvíjejí za pomocí rodičů, z nichž vznikají potomci, kteří při svém vzniku podléhají náhodné mutaci. Mezi všemi jedinci dané populace vzniká konkurence a boj o přežití. Rodiče i potomci se slabými dědičnými vlastnostmi zanikají, a tím uvolňují místo novým, lepším jedincům, kteří se budou dále rozmnožovat, přenášet svoji genetickou náklonnost na další generace a přizpůsobovat se okolí [1].

Evoluční algoritmus pracuje s řadou pojmů z oblasti genetiky, které se používají k popisu struktury algoritmu [1]:

- **Gen** - jednotka genetické výbavy jedince,
- **chromozóm** - řetězec informací, který v sobě nese vlastnosti a chování jedince,
- **jedinec** – představuje jedno řešení dané úlohy,
- **populace** – skupina jedinců v jedné generaci,
- **účelová funkce** – číselné vyjádření kvality jedince.

Cyklus evoluce



Obr. č. 3 Cyklus evoluce u evolučních algoritmů[1].

V prvním kroku jsou definovány všechny potřebné parametry pro řízení běhu algoritmu a jsou stanoveny ukončovací podmínky pro daný algoritmus. Druhý krok je generování první náhodné populace, kde každý jedinec v populaci nese ve svém chromozómu parametry, vytvořené podle počtu argumentů účelové funkce. Poté dochází k ohodnocení kvality jedinců. Všem jedincům se přidělí, za pomoci účelové funkce, jejich hodnota kvality [1].

Výběr rodičů z populace proběhne podle předem stanovených kritérií, kdy snahou je vybírat nejkvalitnější jedince. Jejich následným křížením vzniknou noví potomci, kteří jsou s určitou náhodnou pravděpodobností ovlivněni mutací. Poté proběhne ohodnocení potomků účelovou funkcí [1].

Podle hodnoty účelové funkce proběhne výběr nejlepších jedinců z populace rodičů a potomků. Vybraní jedinci tvoří novou kvalitnější populaci, která plně nahradí populaci starou [1].

Výběr rodičů, mutace, výpočet účelové funkce potomků, výběr z nejlepších potomků a rodičů, a následné vytvoření nová populace, se opakují do té doby, než je splněna některá ukončovací podmínka algoritmu [1].

Použití evolučních algoritmů

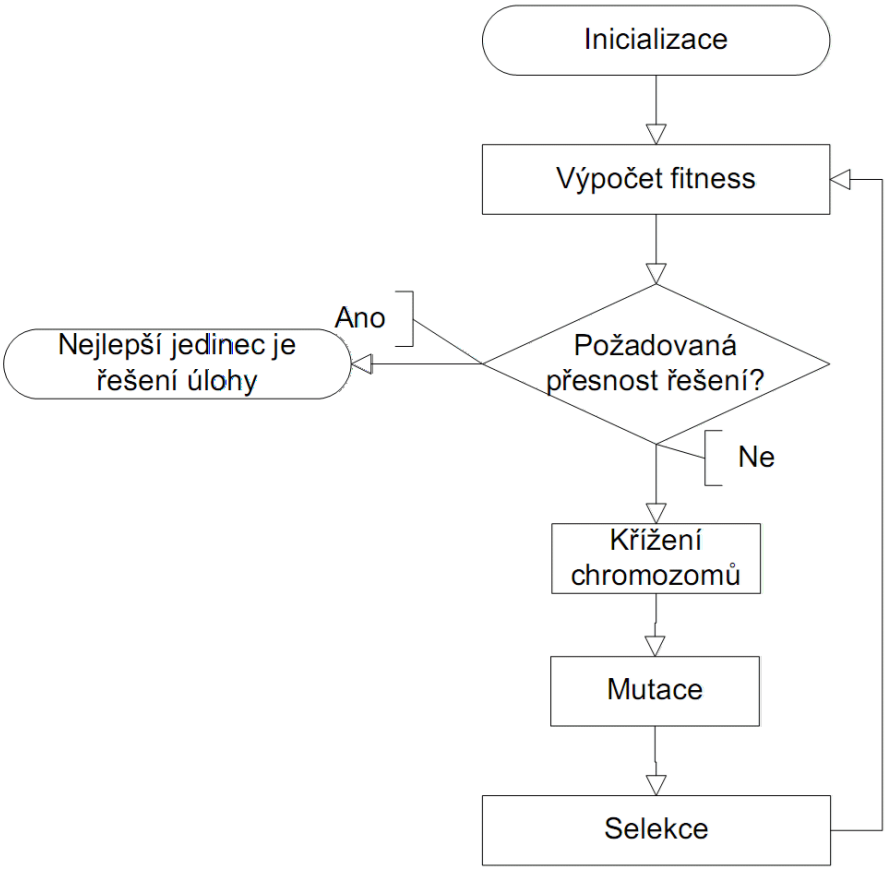
Evoluční algoritmy (EA) jsou vhodné pro řešení složitých optimalizačních problémů, které se velmi obtížně řeší v polynomiálním čase. Významnou charakteristikou evolučních algoritmů je paralelní přístup k řešení úlohy. Evoluční algoritmy pracují s množinou možných řešení, kde každé jednotlivé řešení je popsáno účelovou funkcí, která představuje matematický model problému, který určuje hodnotu kvality řešení. Výsledná hodnota pro jednotlivá řešení je porovnávána s požadovanou hodnotou, a cílem je nalézt hodnoty extrému účelové funkce, tedy její minimum nebo maximum. Toto paralelní prohledávání bodů optimalizační funkce má pozitivní důsledek, že u řešení úlohy je snížená citlivost na počáteční podmínky optimalizace, čímž se snižuje riziko uvíznutí v lokálním extrému [13].

4.1 Genetické algoritmy

Genetické algoritmy (GA) se řadí mezi nejznámější představitele skupiny evolučních algoritmů. Využívají jejich základní myšlenku založenou na napodobení vývoje a učení, které se uplatňují v přírodě. Vzniklý algoritmus lze použít při řešení úloh, které se vyskytují ve složitém, případně i měnícím se prostředí [12].

4.1.1 Popis činnosti genetického algoritmu

Genetické algoritmy se řadí mezi prohledávající algoritmy, které jsou založeny na principech genetiky a mechanismu přirozeného výběru.



Obr. č. 4 Popis činnosti genetického algoritmu [3].

Základem genetických algoritmů je evoluční cyklus, který je opakován, dokud není dosažena požadovaná přesnost účelové funkce.

Prvním krokem je inicializace, neboli vytvoření první populace jedinců. Pro posouzení jednotlivých jedinců se používá klíčové ohodnocení pomocí účelové funkce (fitness), která představuje míru kvality, vhodnosti, síly či reprodukční schopnosti jedince. Jedinci, kteří mají lepší ohodnocení účelové funkce, se častěji podílejí na vytváření nové populace. Ovšem pokud není splněna podmínka požadované přesnosti, jsou na populaci aplikovány operátory selekce, křížení a mutace, pomocí nichž se následně vytvoří nová populace, která se zařadí zpět na počátek evolučního cyklu, kdy se opět vyhodnotí a ověřuje přesnost jednotlivých řešení.

Cílem algoritmu je vytvářet stále silnější jedince, proto se velmi často evoluční cyklus opakuje i vícekrát, obvykle desítky až stovky opakování, než je nalezen jedinec, který má dostatečně přesné ohodnocení, a můžeme jej považovat za optimální řešení daného problému. Tato vlastnost předurčuje algoritmus k použití na řešení optimalizačních problémů, tedy problémů, kdy hledáme nejlepší z možných řešení daného problému.

Obecným problémem u genetických algoritmů je určení okamžiku, kdy je možné zastavit výpočet. Při snaze o co nejvyšší přesnost řešení může nastat situace, kdy běh algoritmu bude probíhat neúměrně dlouho, aniž by bylo nalezeno řešení s minimální požadovanou hodnotou účelové funkce. Proto se používá omezení délky výpočtu na předem stanovených *n* cyklů nebo se využívá možnosti ukončit výpočet, pokud dojde k zastavení růstu maximální hodnoty účelové funkce v populaci. Toto předčasné zastavení růstu naznačuje, že velmi pravděpodobně algoritmus uvázl v lokálním extrému.

4.1.2 Kódování

Způsob kódování genů do chromozómu je často přizpůsoben konkrétnímu řešení. Při kódování se vytvoří řetězec, který je nejčastěji reprezentován v binární podobě, kdy každý gen nabývá pouze hodnoty 0 a 1. Pravidlem je dodržování stejné délky řetězce při řešení optimalizačního problému [1].

Chromozóm A	0	1	0	1	0	1	1	0
Chromozóm B	1	1	0	1	1	0	0	1

Obr. č. 5 Schéma binárního kódování.

Ovšem binární reprezentace jedince není jediná možná podoba, lze také použít celočíselné kódování nebo kódování pomocí reálných čísel. Poté je každý představitel řetězce umístěn na danou pozici v řetězci.

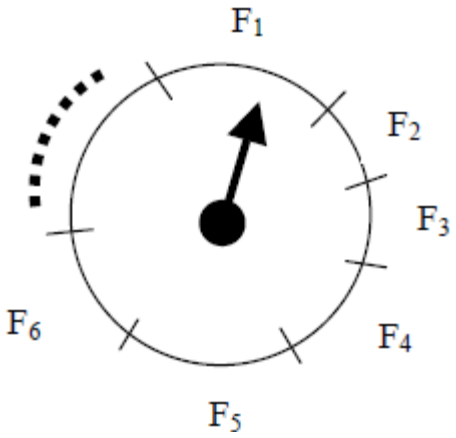
Chromozóm A	1	2	3	4	5	6	7	8
Chromozóm B	6	8	4	7	1	5	3	2

Obr. č. 6 Schéma celočíselného kódování.

4.1.3 Selekce

Selekce slouží k výběru rodičů z populace. Snahou je, aby jedinci nové populace měli co nejvyšší hodnotu účelové funkce. K tomu účelu se vybírají nejlépe hodnocení jedincí, kteří budou předávat své vhodné geny potomkům. Ovšem nelze striktně říci, že nejlépe hodnocení jedinci, jsou ti jediní, kteří mají být vybráni. Aby nedošlo ke konvergenci k lokálnímu extrému, je vhodné vybírat i mezi jedinci s horším ohodnocením, kteří ale zachovají různorodost populace [1].

Ruletová selekce se používá k přerozdělení účelové funkce do vhodnosti rodiče. Jedná se o náhodný výběr, který zvýhodňuje jedince s vyšší hodnotou účelové funkce. Lze si jej představit jako ruletové kolo, kde celková výseč představuje sumu všech ohodnocení jedinců, a poté jsou jednotlivé výseče přerozděleny podle odpovídající velikosti vypočítaných hodnot jedinců [1].



Obr. č. 7 Schéma ruletové selekce [1]

Algoritmus výběru spočívá ve výpočtu sumy S všech účelových hodnot v populaci. V dalším kroku je generováno náhodné číslo r z intervalu $(0, S)$. Následně je vybrán do nové populace jedinec, který má větší hodnotu účelové funkce než náhodné číslo r [1].

Pořadová selekce zamezuje dominanci silných jedinců nad slabými, které se budou jen velmi obtížně podílet na vytvoření nových potomků. Algoritmus spočívá v přerozdělení, kdy jsou jedinci seřazeni podle hodnot účelové funkce, a je jim přidělen stejný úsek pro možný výběr, čímž se stanou rovnoprávnými s podobnou šancí na výběr [14].

Turnajová selekce je metoda, při které se vybírá skupina náhodných jedinců, kteří se utkají v turnaji. Do nové populace postupuje vítěz, popřípadě vítězové, kteří mají z vybrané

skupiny nejlepší hodnocení účelové funkce. Hlavní výhodou této metody je její rychlost, a zároveň nevyžaduje seřazení populace [14].

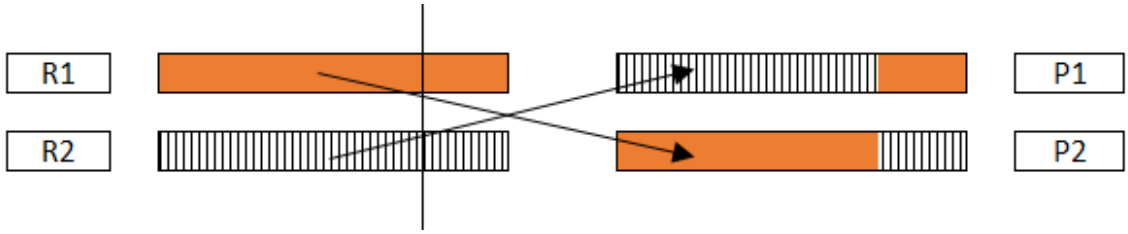
Elitismus zaručuje, že nedojde ke ztrátě nejlepšího řešení, ke které může dojít například při výběru ruletovou nebo pořadovou selekcí. Princip spočívá v tom, že již na začátku selekce se vybere jedinec s nejlepší hodnotou účelové funkce, a ten se automaticky umístí do nové populace [1].

4.1.4 Křížení a mutace

Po úspěšné selekci dvou jedinců, kteří nyní představují rodiče, přichází jejich křížení, čímž dochází k vytvoření potomků, kteří jsou následně podrobeni náhodné mutaci.

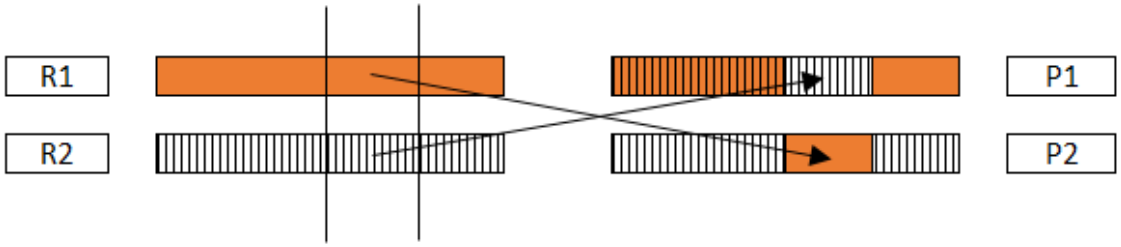
Křížení představuje vzájemnou výměnu části chromozómu mezi rodiči. Křížení je silně závislé na použité metodě kódování, neboť pracuje přímo s chromozómy a geny. Pravděpodobnost křížení, které určuje, jak často se křížení bude provádět, se volí obvykle mezi 60-95% [1].

Jednobodové křížení představuje nejjednodušší metodu, kdy se překříží chromozómy v jednom náhodně vybraném bodě.



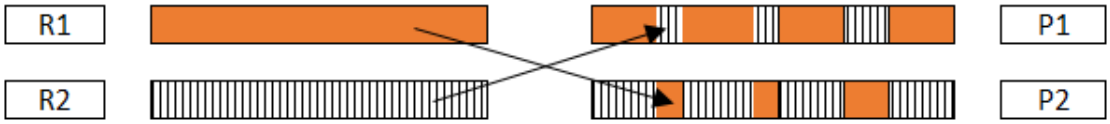
Obr. č. 8 Schéma jednobodového křížení.

Dvoubodové křížení je metoda, kdy se chromozóm rozdělí ve dvou bodech, přičemž se mezi rodiči vymění vybraný úsek mezi těmito body.



Obr. č. 9 Schéma dvoubodového křížení.

Vícebodové křížení spočívá ve výběru několika náhodně vybraných úsecích chromozómů rodičů, které jsou poté vloženy do nového potomka.



Obr. č. 10 Schéma vícebodového křížení.

Mutace přichází na řadu po křížení. Princip mutace spočívá v tom, že s velmi malou pravděpodobností se mění, obvykle náhodně, jednotlivé hodnoty genů chromozómů nových potomků. Pravděpodobnost mutace se volí obvykle pod 1%. Smyslem mutace je zvyšování různorodosti populace a prevence uváznutí v lokálním extrému [1].

Binární mutace - dochází ke změně genů z 1 na 0 nebo opačně.

Výměnná mutace - kdy se na náhodně zvolených pozicích dva geny vymění.

Posuvná mutace - náhodně zvolen gen a ten je posunut na náhodně vygenerovanou pozici.

Chromozóm před mutací	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**
Chromozóm po mutaci	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**

Obr. č. 11 Schéma binární mutace chromozómu.

4.1.5 Ukončovací kritéria

U genetických algoritmů se využívá několika způsobů ukončovacích kritérií, které lze mezi sebou vzájemně kombinovat [1]:

- **účelová funkce nejlepšího jedince** dosáhla požadované přesnosti řešení. Tento je vybrán jako konečné řešení daného problému,
- **počet generací** udává maximální počet generací, při kterých je prováděn výpočet. Po překročení stanovených počtů generací algoritmus končí a je vybrán aktuálně nejlepší jedinec, jako konečné řešení daného problému,
- **neproběhla změna u nejlepší hodnoty účelové funkce za stanovený počet generací po sobě**. Jedná se pravděpodobně o uvíznutí v globálním nebo lokálním extrému,
- **časové ukončení**, kdy je stanoven maximální čas výpočtů a po jeho překročení je vybrán aktuálně nejlepší jedinec jako konečné řešení daného problému.

4.2 Diferenciální evoluce

Diferenciální evoluce (DE) je jedním z evolučních algoritmů, který používá reprezentaci reálnými čísly. Autoři Ken Price a Rainer Storm jej poprvé publikovali v roce 1995. Pro své dobré výsledky se, již velmi brzy po zveřejnění, stal populárním pro využití v různých vědních oborech. Základní princip algoritmu využívá čtyř náhodně vybraných jedinců, kteří za pomocí řídících parametrů vytvářejí nové populace [1].

4.2.1 Parametry

- F – mutační konstanta, jejíž hodnota se pohybuje v rozmezí 0-2,
- CR – práh křížení, který určuje, zdali je při vytváření zkušebního vektoru použit prvek z šumového vektoru nebo z aktivního jedince. Hodnoty se mohou pohybovat v rozmezí 0-1. Nastavíme-li CR = 0, aktivní jedinec se bez evoluce zobrazí do nové populace. Při hodnotě CR = 1 bude zkušební jedinec vytvořen pouze ze tří vybraných rodičů, přičemž se algoritmus bude chovat jako při náhodném hledání,
- D – dimenze problému. Jedná se o počet argumentů účelové funkce,
- NP – velikost populace jedinců. Minimální velikost populace je 4, přičemž při nižší udané hodnotě přestává algoritmus pracovat,
- Generace – udává, kolik evolučních cyklů (generací) proběhne [1].

4.2.2 Popis činnosti diferenciální evoluce

Schéma diferenciální evoluce je podobné genetickým algoritmům, s nimiž má několik společných rysů, jako je např. tvorba populace, výběr rodičů, opakování se v generacích, ale rozdílná je při tvorbě potomků, kteří jsou vytváření ze čtyř rodičů a ne ze dvou, jak je tomu u genetických algoritmů.

Prvním krokem u každé generace je stanovení všech parametrů, které se podílejí na chodu algoritmu. Mezi tyto parametry patří F – mutační konstanta, CR – práh křížení, D – dimenze problému, NP – počet jedinců v populaci. V neposlední řadě je třeba určit, zdali se jedinec bude vytvářet z reálných, celočíselných čísel, atd.

Po stanovení parametrů se vytvoří první populace jedinců, vypočítá se jejich hodnota účelové funkce a začne cyklus generace, kdy je postupně z populace vybírán jeden aktivní jedinec, na kterém je aplikován evoluční cyklus.

Evoluční cyklus (Obr. č. 12) spočívá v náhodném vybrání dalších tří jedinců z populace k aktivnímu jedinci. Mezi těmito čtyřmi jedinci se bude postupně uplatňovat mutace a křížení. Jako první se od sebe odečtou dva vybraní jedinci, kteří vzájemně vytvoří diferenční vektor. Všechny parametry diferenčního vektoru se vynásobí mutační konstantou F, která tímto zmutuje diferenční vektor na váhový diferenční vektor. Ten se poté přičte ke třetímu náhodně vybranému jedinci, čímž se vytvoří šumový vektor [1].

Posledním bodem z evolučního cyklu je tvorba zkušebního vektoru, který se určuje ze šumového vektoru a aktivního jedince. Každý parametr zkušebního vektoru se vybírá pomocí prahu křížení CR, a to tak, že jestliže je vygenerované náhodné číslo menší než konstanta CR, pak se do aktuální pozice zkušebního parametru vkládá hodnota z šumového vektoru, pokud tomu tak není, je vložena hodnota z aktivního jedince. Tímto způsobem se postupně získá celý zkušební vektor [1].

Vypočítá se hodnota účelové funkce zkušebního vektoru, a tato hodnota se porovná s hodnotou účelové funkce aktivního jedince. Do nové populace postupuje ten jedinec, který má kvalitnější hodnotu účelové funkce, čímž je zaručeno, že do nové generace postoupil lepší jedinec. Opakovaným výběrem aktivních, a k němu vždy tří náhodných jedinců, se pokračuje až do konce populace. Tím vznikne nová, kvalitnější, generace jedinců.

Při vzniku nové generace se zkoumá, zdali se nesplnila podmínka ukončení algoritmu. Nejčastějším ukončovacím argumentem je vyčerpání stanoveného počtu generací, popřípadě stagnací výše hodnoty účelové funkce nejlepšího jedince v posledních několika generací.

4.2.3 Vybrané strategie diferenciální evoluce

Diferenciální evoluce umožňuje více strategií. Liší se výpočtem šumového vektoru a způsobem vytváření zkušebního vektoru [1].

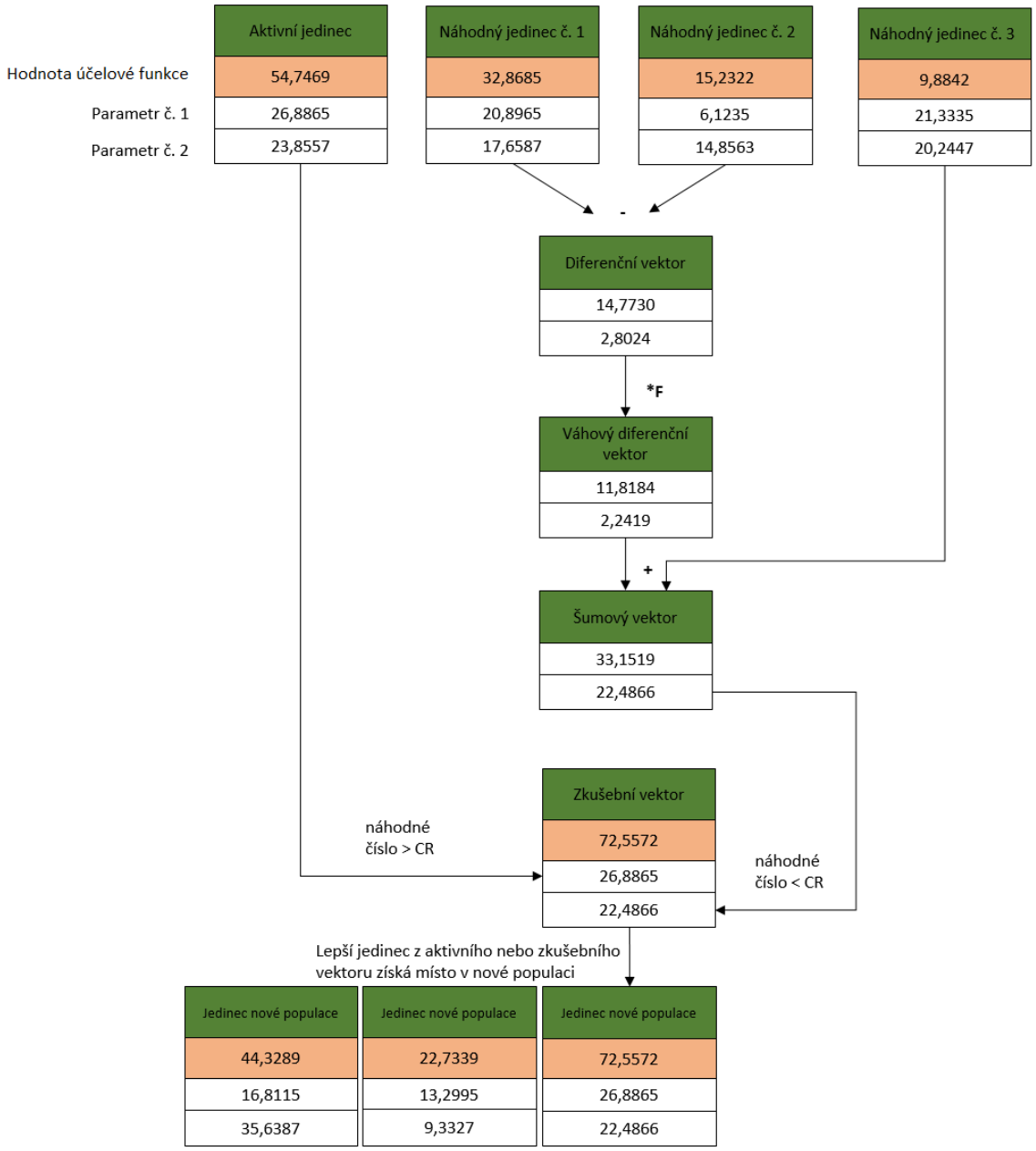
Strategie DE/rand/1/bin: $$v = x_{r1,j}^{G} + F(x_{r2,j}^{G} - x_{r3,j}^{G}) \quad (17)$$

Strategie DE/best/1/bin: $$v = x_{best,j}^{G} + F(x_{r2,j}^{G} - x_{r3,j}^{G}) \quad (18)$$

Strategie DE/rand/2/bin: $$v = x_{r5,j}^{G} + F(x_{r1,j}^{G} + x_{r2,j}^{G} - x_{r3,j}^{G} - x_{r4,j}^{G}) \quad (19)$$

Strategie DE/best/2/bin: $$v = x_{best,j}^{G} + F\left(x_{1,j}^{G} + x_{r2,j}^{G} - x_{r3,j}^{G} - x_{r4,j}^{G}\right) \quad (20)$$

Strategie DE/rand-to-best/1/bin: $$v = x_{i,j}^{G} + \lambda\left(x_{best,j}^{G} - x_{i,j}^{G}\right) + F(x_{r1,j}^{G} - x_{r2,j}^{G}) \quad (21)$$



Obr. č. 12 Princip diferenciální evoluce, verze DE/Rand/1/Bin [19].

4.2.4 Stagnace

O stagnaci vývoje hodnot účelové funkce se hovoří v okamžiku, kdy dojde k zastavení vývoje před nalezením globálního maxima. Částečně se lze vyhnout tomuto jevu zvětšením počtu jedinců v populaci, což má ovšem za následek, že bude provedeno více výpočtů hodnot účelové funkce jedinců, a tím i zpomalení celkové doby nalezení nejkvalitnějšího jedince ze všech populací [1].

4.3 Další vybrané evoluční techniky

Existuje samozřejmě více evolučních technik, než doposud uvedené. Za všechny ostatní je zde stručně představena moderní evoluční technika SOMA (Samo-Organizující se Migrační Algoritmus). Tato metoda sice nebude dále prakticky řešená, ale je zde uvedená jako alternativa ke dvěma, v praktické části, řešeným metodám.

4.3.1 SOMA (Self-Organizing Migrating Algorithm)

Hlavní myšlenka spočívá v rozptýlení jedinců do určitého prostoru funkce (na jejíž části budeme hledat globální extrém) a na jejich částečně řízeném a částečně náhodném pohybu. Nevytváří se tedy evolučně noví jedinci, ale dochází k jejich přemístění ve stavovém prostoru. SOMA je založena na vektorových operacích, kdy jeden evoluční cyklus se zde nazývá migrační kolo, kdy vznikají skupiny jedinců, které se rozpadají a zase spojují a tím se vzájemně ovlivňují [20].

Parametry SOMA [20]:

1. Řídící parametry, které mají vliv na kvalitu běhu algoritmu z hlediska účelové funkce
    - **PathLength** (délka cesty) ϵ (1, 5]. Určuje, jak daleko se zastaví aktivní jedinec od tzv. vedoucího jedince,
    - **Step** (krok) ϵ (0,11;PathLength]. Určuje počet zastavení a vyhodnocení účelové funkce směrem k vedoucímu jedinci,
    - **PopSize** ϵ [10, dáno uživatelem] - kolik jedinců bude tvořit populaci. Min. = 2 přičemž bude SOMA rovna klasickým deterministickým metodám.
    - **PRT** ϵ [0, 1] - Perturbace (rušení) - podle tohoto parametru se tvoří perturbační vektor, který ovlivňuje, zda se aktivní jedinec bude pohybovat přímo k vedoucímu jedinci či nikoliv.
    - **D** parametr - počet argumentů účelové funkce. Určuje dimenzi řešeného problému.
2. ukončovací parametry, které za předem nadefinovaných podmínek algoritmus ukončují:
    - **MinDiv** - zastavovací parametr, který je definován jako nejvyšší přípustný rozdíl nejhoršího a nejlepšího jedince v aktuální populaci.
    - **Migrace** - ukončovací parametr, který udává, kolikrát se populace jedinců přeorganizuje.

II. PRAKTICKÁ ČÁST

5 VYMEZENÍ ŘEŠENÉHO PROBLÉMU

Praktický úkol byl řešen ve firmě Continental Barum s.r.o., která je předním výrobcem plášťů pro osobní a lehké nákladní automobily (PLT) v Evropě, s aktuálně více než 21 miliony vyrobených komerčních plášťů. V celém areálu firmy Continental Barum s.r.o. v Otrokovicích probíhá ucelená výroba PLT plášťů v několika divizích. Hlavní výroba spočívá v mateřské části firmy Continental Barum s.r.o. (CoBa), poté v dceřiné společnosti Continental HT Tyres, s.r.o. (CHTT) a Continental výroba pneumatik, s.r.o. (CVP).

Požadavkem bylo vytvořit efektivní rozvrhování výroby surových plášťů na oddělení konfekce divize Continental Barum s.r.o. Úkolem bylo se zaměřit zejména na efektivní využití strojního zařízení a minimalizovat ztráty na výrobním zařízení lisovny, která je navazující výrobní operací. Lisovna v tomto případě představuje zákazníka konfekce, která se tímto stává jejím dodavatelem.

Jednotlivé výrobky jsou jednoznačně identifikovány pomocí PROSI (PROduction Planning, Scheduling and Information) kódu, který slouží k přehledu o potřebném množství materiálů pro měsíční a denní plán.

Výroba probíhá na 39 výrobních modulech, kdy každý modul představuje dvě strojní zařízení. Na prvním stroji se vyrobí kostra pláště, a na druhém stroji je po vytvarování kostry plášť dokončen uložením nárazníkového prstence s běhounem. Tyto operace se provádějí na různých typech strojů KM a PU firmy Continental. Z pohledu rozvrhování výroby se bere v potaz vždy pouze celý modul, neboť rozhodujícím faktorem je až výroba celého surového pláště. Proto je zde brán jeden modul jako jeden stroj [18].

6 ZADÁNÍ A ŘEŠENÍ PROBLÉMU

Cílem této práce je vytvořit operativní rozvrh výroby surových plášťů pro jeden stroj na 24 hodin. Hlavním kritériem rozvrhu by měla být minimalizace součtu ztrát spojených s nedodržením termínů, přičemž je uplatňována rozdílná priorita úloh. Z charakteru úlohy vyplývá, že se jedná o třídu úloh, které mají permutační charakter, což znamená, že pro *n* úloh je *n!* možných řešení.

6.1 Řešení problému pomocí genetických algoritmů

Vzhledem k tomu, že permutační rozvrhovací úlohy patří do třídy úloh NP-těžké, tudíž je velmi obtížné řešit v polynomiálním čase, uplatňují se při jejich řešení algoritmy založené na metaheuristických přístupech, mezi něž se řadí i genetické algoritmy. Řešení je vypracováno v jazyce Visual Basic for Applications pro aplikaci Microsoft Excel.

6.1.1 Nastavení parametrů genetického algoritmu

Jako první je nutné definovat parametry genetického algoritmu:

- velikost populace – určuje, kolik jedinců bude v jedné populaci,
- počet populací – parametr, který udává maximální počet populací,
- pravděpodobnost křížení *prob_crossover* – udává pravděpodobnost, s jakou se budou mezi sebou jedinci křížit,
- pravděpodobnost mutace *prob_mutation* - udává pravděpodobnost, s jakou se bude provádět mutace,
- stanovení vyhovující hodnoty účelové funkce – hodnota, podle níž lze určit, zdali je nalezené řešení problému vyhovující.

6.1.2 Vstupní data pro rozvrhování surových plášťů na jednom stroji

Tato vstupní data slouží k samotnému výpočtu účelové funkce pro všechny výrobky na jednom stroji:

- název (kód) výrobku na stroji – slouží pro přesnou identifikaci výrobku v rozvrhu,
- čas výroby jednoho kusu výrobku na daném stroji,
- počet kusů výrobku odebraných ze systému – stanovení počtu kusů výrobku, které se budou odečítat z aktuálního počtu výrobků za stanovený čas,
- salda výrobků – hodnota udávající stav počtu kusů pro jednotlivé výrobky.

6.1.3 Omezující parametry

Poté je vhodné stanovit omezující parametry:

- maximální počet vyrobených kusů daného výrobku – omezující podmínka, která určuje maximální počet kusů výrobku, který vstupuje do rozvrhu. Maximální hodnota se vypočítá podle vzorce (22), kde figurují nastavitelné hodnoty maximálního časového salda (např. 12 hodin), které se násobí počtem kusů odebraných ze systému, tedy vylisovaných za 1 hodinu, a počtem forem rozměru,
- minimální počet kusů výrobku - omezující podmínka, která určuje, zdali je ještě dostatečný počet kusů výrobku (vzorec 25).

6.1.4 Vytvoření jedinců první populace

Prvním krokem je vytvoření první náhodné populace, která se skládá z několika jedinců. Tito jedinci jsou reprezentováni celočíselným kódováním, a jejich jednotlivé geny jsou tvořeny znaky, jejichž počet je určen proměnnou `number_Char`. Tyto geny mohou představovat libovolné hodnoty. Konkrétně pro zpracovávané řešení budou použity vždy tři znaky, představující hodnoty od 000 až do vypočítaného maximálního počtu vyrobených kusů jedné dávky pro konkrétní rozměr, s možným maximem 999 kusů. Maximální počet kusů se vypočítává pro každý výrobek zvlášť, a to podle vzorce (22):

```
max_item(prosi) = mold(prosi) * piecesPerHour * balanceTime
```
(22)

kde položka `mold` představuje počet forem rozměru na lisovně, `piecesPerHour` počet kusů vylisovaných za jednu hodinu a `balanceTime` je nastavitelná hodnota, která představuje na kolik hodin je možné mít vyrobenou zásobu surových plášťů.

Vzorec (22) představuje prakticky omezující podmínku, která zaručuje, že se pro daný výrobek nevyrobí příliš velké množství surových plášťů, zejména s ohledem na jejich přepravní a uskladňovací maximální kapacity před samotným lisováním.

Každý jedinec z populace má tedy stejnou délku, která je, spolu s délkou každého genu, také určena počtem specifických výrobků, v tomto případě rozměrů na stroji. Každý rozměr je uveden v rozvrhu třikrát, neboť by se měl vyrábět každou směnu, což představuje právě třikrát za 24 hodin. Pořadí rozměrů je uloženo v datové struktuře s názvem `prosi`.

První pořadí rozměrů se určuje pomocí priority, která je vypočítána podle vzorce (23).

```
Priority(prosi) = mold(prosi) / balance(prosi)
```
(23)

Vzorec (23) představuje poměr počtu forem výrobku na lisovně `mold(prosi)` a výchozího stavu počtu kusů výrobku `balance(prosi)`.

Tedy čím méně je vyrobených surových plášťů na jednu formu, tím získává rozměr větší prioritu, a bude se dříve vyrábět.

Tímto je zaručeno, že se určí pořadí výrobků pro první směnu, které minimalizuje počáteční prostoje v daném rozvrhu. Stejné pořadí zůstává zachováno i pro další dvě směny, zejména z důvodu snadnějšího plánování přípravných materiálů.

Zde je příklad výpočtu délky jedince na stroji, kde je 5 rozměrů. Výsledek je počítán jako 5 * 3 * 3 = 45 znaků. První číslo z výpočtu představuje počet rozměrů `number_Products`, druhé počet opakování za 24 hodin a poslední číslo je počet znaků genu `number_Char`, v našem případě 3 znaky.

240060060060003600000600601001200600600602 40

Nyní je znám počet výrobků a jejich pořadí, kdy u každého z nich je náhodně určen počet vyrobených kusů v jedné dávce, a tedy po oddělení jednotlivých genů lze vyčíst z chromozómu jedince:

240 060 060 060 000 360 000 060 060 100 120 060 060 060 240

že v tomto případě se bude první rozměr vyrábět na 1. směně v počtu 240 kusů, na druhé 360 kusů a na třetí směně 120 kusů. Ovšem druhý rozměr se bude vyrábět pouze dvakrát, protože pro první směnu je rozepsáno 60 kusů, pro druhou 0 kusů a pro třetí směnu 60 kusů. Nula na druhé směně je proto, že pravděpodobně není nutné tento rozměr v danou dobu vyrábět, protože je v tomto čase dostatek surových plášťů. O tom, zdali právě tento vygenerovaný rozvrh je optimální, rozhoduje hodnota účelová funkce a nastavené ukončovací kritéria.

6.1.5 Výpočet účelové funkce

Hodnota účelová funkce, nazývaná také jako cost value (CV), hodnotí, jak je dobré konkrétní řešení v populaci. U každého jedince v populaci se pomocí CV určuje jeho kvalita, a tím je možné jednotlivé jedince mezi sebou porovnat.

Pro výpočet hodnoty účelové funkce je nutné znát nebo vypočítat níže uvedené parametry, a to pro každý rozměr na stroji:

- potřebný čas na vyrobení jednoho výrobku na stroji `time_Item`,
- výchozí počet surových plášťů `quantity(prosi) = balance(prosi)`,
- výrobní dávka `pieces(prosi)`, která je určena pro rozměr z chromozómu jedince,
- počet kusů vylisovaných za hodinu:

  ```
  piecesPerHour(prosi) = mold(prosi) * curing_time(prosi)
  ```
  (24)

  kde `mold(prosi)` představuje počet forem rozměru a `curing_time(prosi)` je čas lisování jednoho kusu výrobku,
- minimální počet kusů rozměru

  ```
  min(prosi) = mold(prosi) * curing_time(prosi)* minTime
  ```
  (25)

  kde `mold(prosi)` představuje počet forem rozměru, `curing_time(prosi)` je čas lisování jednoho kusu rozměru a `minTime` představuje čas, který určuje, jaká je časová rezerva mezi začátkem výroby a nulovým počtem kusů surových plášťů,
- určení všech časů začátku výroby rozměrů v průběhu 24 hodin `time_Production`.

Pro určení všech časů začátku výrobních dávek jednotlivých rozměrů se nejdříve použije vzorec (26), pomocí kterého se vypočítá čas výroby první výrobní dávky prvního rozměru:

```
time_Production_Item = time_Item(prosi) * pieces(prosi)
```
(26)

kde `time_Item(prosi)` je potřebný čas na vyrobení jednoho kusu na stroji a `pieces(prosi)` je počet kusů, které jsou reprezentovány jednotlivými geny v chromozómu jedince.

Výpočet podle vzorce (26) se přičte k výchozímu času nula a poté tento výpočet probíhá pro všechny rozměry a dávky na stroji. Takto vypočítané časy se postupně k sobě přičítají podle vzorce (27) a tímto je vždy stanoven aktuální výrobní čas od začátku výroby, který je ihned přiřazen k právě počítanému rozměru. Tímto je určen přesný čas zahájení výroby u každého rozměru a při každém jeho zahájení výroby.

```
time_Production = time_Production + time_Production_Item
time_Production (prosi) = time_Production
```
(27)

Nyní může začít výpočet samotné účelové funkce jedince, kdy se postupně procházejí všechny výrobky na stroji, k nimž jsou vypočítány všechny potřebné pomocné hodnoty, které jsou postupně využívány k výpočtu v cyklu 24 hodin, a to v iteraci po jedné hodině.

Cyklus 24 hodin je opakován pro každý rozměr zvlášť, a každou iteraci, tedy každou hodinu, proběhnou následující kroky:

1. z aktuálního množství surových plášťů konkrétního rozměru `quantity(prosi)` se odečte počet kusů, které lisovna vylisuje za 1 hodinu:

   `quantity(prosi) = quantity(prosi) – piecesPerHour(prosi)` (28)

   kde `piecesPerHour(prosi)` představuje počet vylisovaných kusů výrobků za jednu hodinu,
2. pokud se shoduje aktuální čas v cyklu s časem, kdy se má výrobek vyrábět `time_Production(prosi)`, tak proběhne výpočet podle vzorce (29).

   `quantity(prosi) = quantity(prosi) + pieces(prosi)` (29)

   kde `quantity(prosi)` je počet aktuálních kusů a `pieces(prosi)` je přidělená výrobní dávka.

   Tímto výpočtem vznikne poněkud zavádějící číslo, které je nadhodnocené a aktuálně neukazuje reálný počet kusů. Je to z toho důvodu, že je v jeden okamžik přidělena celá výrobní dávka, a ne postupně, jak je tomu v reálné výrobě. V tomto případě to nevadí, neboť výroba surových plášťů je mnohonásobně rychlejší (cca 10x) než jejich lisování a hlavním rozhodujícím prvkem je minimální hodnota aktuálního počtu kusů po skončení výrobní dávky, což je nyní splněno. Pokud by tento předpoklad neplatil, bylo by nutné výrobní dávku rozdělit poměrově do více iterací.
3. pokud je aktuální počet kusů `quantity(prosi)` menší nebo roven minimálnímu počtu kusů `min(prosi)`, je proveden výpočet podle ukázky zdrojového kódu č. 1:

```
if quantity(prosi) <= min(prosi) then
      if time(i) < 8 then downTime8 = downTime8 + 1
      if time(i) =>8 and time(i)<=16 then downTime16=downTime16 + 1
      if time(i) > 16 then downTime24 = downTime24 + 1
end if
```

Zdrojový kód č. 1 Výpočet prostojových hodin podle času od zahájení výroby.

kde `time(i)` je aktuální iterace (hodina výroby), `downTime8` jsou prostojové hodiny za prvních 8 hodin, `downTime16` jsou prostojové hodiny od 8 do 16 hodin a `downTime24` jsou prostojové hodiny nad 16 hodin od stanoveného začátku.

Celý cyklus, pro aktuálně počítaný rozměr, končí po 24 iteracích (hodinách), kdy jsou již vypočítány celkové časové prostojové hodiny rozměru ve třech intervalech, které jsou poté všechny vynásobeny počtem forem rozměru na lisovně `mold(prosi)`, což určí celkový čas prostojů rozměru `downTime(prosi)`, kdy bylo málo surových plášťů daného rozměru pro lisování.

```
downTime8 = downTime8 * mold(prosi)
downTime16 = downTime16 * mold(prosi)
downTime24 = downTime24 * mold(prosi)
downTime(prosi) = downTime8 + downTime16 + downTime24
```

Zdrojový kód č. 2. Celkový výpočet prostojových hodin podle času od zahájení výroby.

kde `downTime(prosi)` je celkový počet prostojových hodin rozměru, `downTime8`, `downTime16`, `downTime24` jsou prostojové hodiny podle určení času výroby od začátku výpočtu a `mold(prosi)` je počet forem rozměru.

Dále se vypočítá celková penalizační hodnota rozměru pro účelovou funkci `bad_Time(i)`. Tato hodnota má přímý vliv na celkovou hodnotu účelové funkce `fitness` (Zdrojový kód č. 3., str. 40). Penalizační hodnota rozměru se vypočítá podle vzorce (30):

```
bad_Time(i) = downTime8*50 + downTime16*20 + downTime24*5
```
(30)

kde `downTime8, downTime16, downTime24` jsou prostojové hodiny podle času od začátku výroby a číslice 50, 20 a 5 jsou stanoveny podle následujícího pravidla, kdy jsou celkové prostojové hodiny pro prvních 8 hodin `downTime8` navýšeny o penalizační hodnotu 2,5 krát vyšší než `downTime16` pro rozmezí 8 až 16 hodin, a 10 krát vyšší než `downTime24` pro 16 a více hodin, to vše počítáno od začátku výroby. Je to z toho důvodu, že lze na pozdější nedostatek surových plášťů reagovat, např. výpomocí na jiném stroji, což během prvních 8 hodin není obvykle možné nebo je přinejmenším obtížné rychle reagovat.

Dalším krokem výpočtu účelové funkce je přechod na další rozměr na stroji, kdy se celý cyklus opakuje od bodu číslo 1, a to vždy, dokud nejsou vypočítány všechny hodnoty i pro poslední rozměr na daném stroji.

Celková hodnota účelové funkce jedince `fitness` je poté vypočítána na základě všech prostojů u všech rozměrů přidělených na stroji:

```
For i = 1 to number_Products
      fitness = fitness + bad_Time(i)
Next i
```

Zdrojový kód č. 3. Výpočet celkové hodnoty účelové funkce.

6.1.6 Ukončovací kritéria pro první populaci

Pokud je nalezen rozvrh bez prostojů, tedy s minimální hodnotou účelové funkce, ukončí se výpočet algoritmu. Naopak, pokud není dosažena stanovená minimální hodnota účelové funkce, je nutné vytvořit pomocí turnajového výběru, křížení a mutace novou populaci, a opakovat celý výpočet pro novou populaci.

6.1.7 Selekce

V případě, že se u první populace neukončí evoluční vývoj, pokračuje se vytvářením potomků ze dvou vybraných rodičů. Náhodně se vyberou dva jedinci z populace za pomocí funkce:

```
Function selectGene()
   selectGene = Int(((population_size - 1) - 0 + 1)*Rnd + 0)
End Function
```

Zdrojový kód č. 4. Náhodná selekce jedince z populace.

6.1.8 Křížení

Na vybraných jedincích je aplikováno křížení, které je zahájeno vygenerováním náhodného čísla *rand_crossover*, které je porovnáno se stanovenou hodnotou pravděpodobnosti křížení *prob_crossover*. Pokud je náhodné číslo menší než pravděpodobnost křížení, je provedeno samotné křížení.

Je zvoleno náhodné číslo *cross_point*, které je vždy násobkem počtem znaků `number_Char` hodnoty, která představuje počet znaků reprezentujících počet kusů, které se mají vyrábět. V tomto případě je počet znaků 3.

V bodě *cross_point* je poté provedeno křížení obou chromozómů rodičů.

```
'Cross genes at random spot in strings
new_gene_1 = Mid(population.Item(gene_1), 1, cross_point - 1) & Mid_
(population.Item(gene_2), cross_point, number_of_items * number_Char)
new_gene_2 = Mid(population.Item(gene_2), 1, cross_point - 1) & Mid_
(population.Item(gene_1), cross_point, number_of_items * number_Char)
```

Zdrojový kód č. 5. Křížení dvou chromozómů rodičů (GA).

Pokud je náhodné číslo větší než pravděpodobnost křížení, postupují do nové generace automaticky oba náhodně zvolení rodiče.

6.1.9 Mutace

Mutace jedince následuje jako další krok po křížení, kdy je vygenerováno náhodné číslo *rand_mutation*, které je porovnáno s nastavenou hodnotou pravděpodobnosti mutace *prob_mutation*. Pokud je náhodné číslo menší než pravděpodobnost mutace, je provedena samotná mutace, jinak postupuje jedinec dále beze změny.

Princip mutace je takový, že se náhodně vyberou dva body z chromozómu jedince:

```
mut_point = (Int(((number_of_items - 1)- 1 + 1) * Rnd + 1)) * number_Char
mut_point2 = (Int((number_of_items / 2 -1)-1 + 1) * Rnd + 1)* number_Char
```

Zdrojový kód č. 6. Výběr dvou náhodných bodů pro mutaci (GA).

Tyto dva náhodné body určují pozici v chromozómu pro dvě výrobní dávky (počet kusů, které se mají vyrábět), a jedna z těchto dvou vybraných výrobních dávek zmutuje v druhou, to znamená, že ji na její pozici nahradí, čímž vznikne zmutovaný jedinec:

```
mut_gene = breed_population.Item(breed_population.Count - 1)
pieces_temp_1 = Mid(mut_gene, mut_point + 1, number_Char)
pieces_temp_2 = Mid(mut_gene, mut_point2 + 1, number_Char)
pieces_temp_1 = pieces_temp_2
left_text = Left(mut_gene, mut_point)
right_text = Right(mut_gene, Len(mut_gene) - mut_point - number_Char)
breed_population.Item(breed_population.Count - 1) = left_text & _
pieces_temp_1  & right_text
```

Zdrojový kód č. 7. Mutace jedince u genetického algoritmu.

6.1.10 Turnajový výběr

Po křížení a mutaci je vytvořen potomek, jehož kvalita je ohodnocena účelovou funkcí, poté je tato hodnota porovnána s hodnotami účelové funkce obou rodičů. Pokud je potomek kvalitnější, nahradí jednoho či oba rodiče v nové populaci.

```
If temp_fitness_offspring < temp_fitness_parent Then
    population.Item(parent) = breed_population.Item(offspring)
    fitness.Item(parent) = temp_fitness_offspring
End If
```

Zdrojový kód č. 8 Turnajový výběr u genetického algoritmu.

6.1.11 Ukončovací kritéria

1. Pokud je nalezen rozvrh bez prostojů, tedy s minimální hodnotou účelové funkce, ukončí se výpočet algoritmu. Naopak, pokud není dosažena stanovená minimální hodnota účelové funkce, je nutné vytvořit pomocí turnajového výběru, křížení a mutace novou populaci, a opakovat celý výpočet pro novou populaci,
2. jestliže je vyčerpán stanovený počet populací, tak se ukončí výpočet algoritmu.

6.1.12 Vizualizace rozvrhu

Po vytvoření rozvrhu výroby jednotlivých rozměrů na stroji je tento rozvrh zobrazen ve dvou tabulkách (Obr. č. 13 a Obr. č. 14). Na obrázku č. 13 lze vidět, jak byly rozměry na stroji uspořádány a jaké jsou jejich výrobní dávky i s časy zahájení výroby. Barevně je také rozlišeno, pro jakou směnu jsou přiděleny rozměry a jejich začátky výroby. Modrá je v tomto případě odpolední směna, zelená noční směna a červená představuje ranní směnu následujícího dne. Bílá značí, že začátek výroby rozměru je více než 24 hodin od začátku výpočtu a není již zahrnut do výpočtu účelové funkce. Je zde pouze jako informace, v jakém rozměru bude výroba pokračovat.

5200002833	**400**	13:45:00
5200000390	**200**	19:56:37
5200002079	**200**	23:02:25
5200002833	**380**	2:08:14
5200000390	**220**	8:01:15
5200002079	**160**	11:25:39
5200002833	**160**	13:54:17

Obr. č. 13 Ukázka vizualizace rozvrhu.

Na obrázku č. 14 je znázorněna ukázka vizualizace rozměrů uvedených na obrázku č. 13 a jejich pohybu počtu kusů u jednotlivých rozměrů na prvních 10 hodin. V prvním sloupci je uveden rozlišovací PROSI kód rozměru, ve druhém je počet kusů, které lisovna vylisuje za 1 hodinu, a počínaje třetím sloupcem jsou udávány počty kusů v daném čase, kdy se postupně odečítají vylisované kusy a přičítají vyrobené dávky v určeném čase. Zeleně podbarvené políčka značí zahájení výroby rozměrů mezi daným časem a časem předešlým, kdy se přiřadí plná výrobní dávka. Z obrázku č. 14 lze tedy např. vyčíst, že rozměr 5200000390 začal vyrábět mezi 19:45 a 20:45, což dokazuje i výrobní čas tohoto rozměru 19:56:37, který je přidělen tomuto rozměru na obrázku č. 13. Počet vylisovaných kusů za 1 hodinu se vypočítává podle vzorce (24), výpočet aktuálních kusů podle vzorce (28) a (29).

Rozměr	Vylisovaných za 1 hod	13:45	14:45	15:45	16:45	17:45	18:45	19:45	20:45	21:45	22:45	23:45
5200002833	31,5	134	503	471	440	408	377	345	314	282	251	219
5200000390	18,5	200	181	163	144	126	107	89	270	252	233	215
5200002079	14,0	184	170	156	142	128	114	100	86	72	58	244

Obr. č. 14 Ukázka vizualizace rozvrhu pro prvních 10 hodin a vývoj počtu kusů v čase.

6.1.13 Kontrola výsledků GA rozvrhu pro daný stroj

Pro ověření, jestli algoritmus vytváření rozvrhu nalezne pokaždé stejný výsledek nebo minimálně podobný výsledek, byl vybrán stroj, na kterém se mají vyrábět tři rozměry. Potřebné parametry výrobků jsou uvedeny v tabulce 1.

Tab. 1 Vstupní parametry surových plášťů – kontrola rozvrhu.

Rozměr PROSI kód	Počet kusů	čas výroby (s) / 1 ks	forem 1.sm	forem 2.sm	forem 3.sm	Priorita	lis ks/hod
5200002833	134	0,56	6	6	6	0,044	5,2
5200000390	200	0,56	4	4	4	0,020	4,6
5200002079	184	0,56	3	3	3	0,016	4,7

Tab. 2 Nastavení parametrů genetického algoritmu – kontrola rozvrhu.

Velikost populace	60
Počet generací	25
Pravděpodobnost křížení	0,8
Pravděpodobnost mutace	0,03
Maximální čas salda (hod)	24
Minimální čas salda (hod)	4

Nastavení parametrů GA (Tab. 2) bylo zvoleno po několika experimentech na hodnotu 0.8, což znamená, že s 80% pravděpodobností dojde ke křížení. Mutace bude prováděna ve velmi malém procentu výpočtů.

Cílem testování byla kontrola výpočtu rozvrhu, a zdali dává podobné výsledky i při více nezávislých spuštěních. Zadání bylo stále stejné, jen výpočet se spustil celkem třicetkrát po sobě, a vždy se stejnými výchozími podmínkami. V tabulkách 3 a 4 představují řádky číslo generace, kterých je v tomto případě 25, a sloupce představují pořadové číslo spuštění. V tabulce 3 je uvedeno prvních 15 a v tabulce 4 je druhých 15. Uvedené hodnoty představují hodnotu účelové funkce, která reprezentuje aktuálního nejlepšího jedince. Vzhledem k tomu, že je hledáno minimum funkce, je nejkvalitnější hodnota prezentována nulou. Vyšší velikost hodnoty účelové funkce znamená rostoucí chybu v rozvrhu.

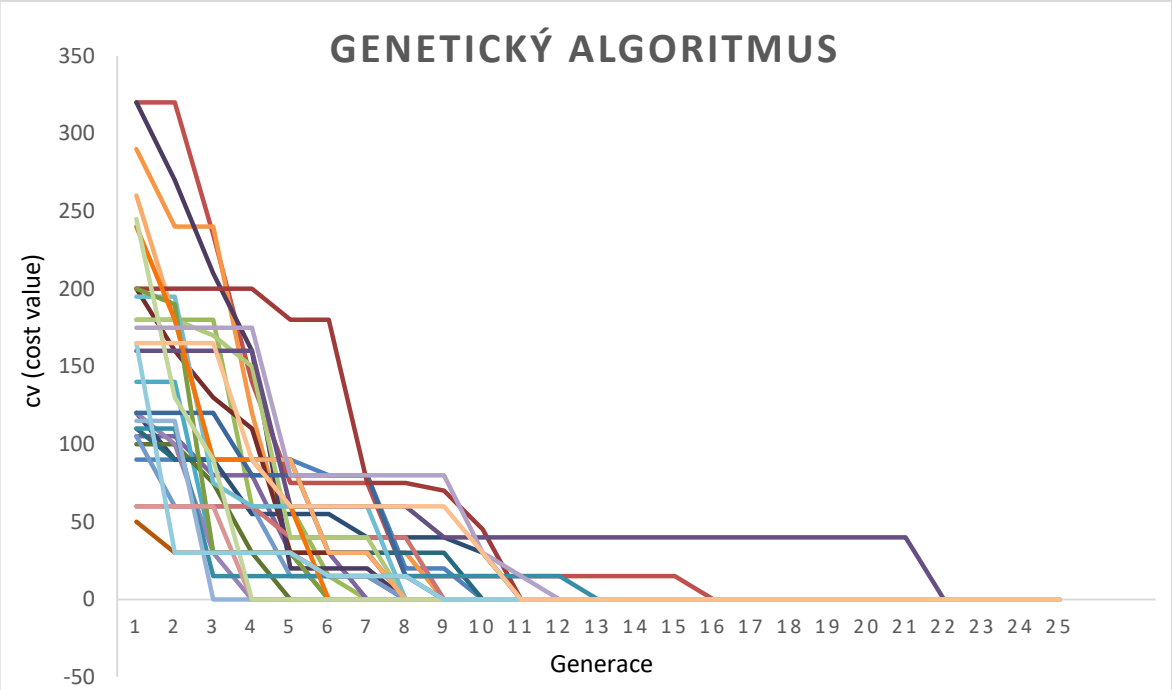
Tab. 3 Kontrolní výpočty genetického algoritmu pro prvních 15 výpočtů.

Počet generací \ Počet měření	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	90	320	180	105	140	290	120	200	100	320	110	50	105	60	180
2	90	320	180	105	140	240	90	160	100	270	90	30	60	60	180
3	90	235	180	80	30	240	90	130	75	210	90	30	60	60	170
4	90	140	60	80	30	120	55	110	30	160	90	30	60	60	150
5	90	75	60	30	30	30	55	30	0	20	90	30	15	40	40
6	80	75	15	30	0	30	55	30	0	20	30	15	15	40	40
7	80	75	0	0	0	30	40	30	0	20	30	15	15	40	40
8	20	15	0	0	0	30	40	0	0	0	30	0	0	40	0
9	20	15	0	0	0	0	40	0	0	0	30	0	0	0	0
10	0	15	0	0	0	0	30	0	0	0	0	0	0	0	0
11	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tab. 4 Kontrolní výpočty genetického algoritmu pro dalších 15 výpočtů.

Počet generací \ Počet měření	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	120	195	260	120	200	200	160	110	240	115	60	245	175	165	165
2	100	195	180	120	200	190	160	110	180	115	60	130	175	30	165
3	30	75	90	120	200	30	160	15	90	0	60	90	175	30	165
4	0	60	90	80	200	30	160	15	90	0	0	0	175	30	90
5	0	60	90	80	180	30	60	15	60	0	0	0	80	30	60
6	0	60	30	80	180	0	60	15	0	0	0	0	80	15	60
7	0	60	30	80	75	0	60	15	0	0	0	0	80	15	60
8	0	0	0	15	75	0	60	15	0	0	0	0	80	15	60
9	0	0	0	0	70	0	40	15	0	0	0	0	80	0	60
10	0	0	0	0	45	0	40	15	0	0	0	0	30	0	30
11	0	0	0	0	0	0	40	15	0	0	0	0	15	0	0
12	0	0	0	0	0	0	40	15	0	0	0	0	0	0	0
13	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Pro účely kontroly výsledků rozvrhu nebylo využito ukončovací kritérium, které ukončí výpočet po nalezení nulové hodnoty účelové funkce. Poté z obou tabulek vyplývá, že hledané globální minimum bylo nalezeno ve všech třiceti případech. Chybová hodnota účelové funkce v první generaci byla, podle předpokladů, ve všech třiceti výpočtech nejvyšší ze všech 25 generací. I u druhé generace stále nelze vyčíst nějaké významnější zlepšení, které se projevuje obvykle ve třetí až čtvrté generaci. Od páté generace lze, podle výsledků v tabulkách 3 a 4, usoudit, že dochází ke zlepšení hodnot účelové funkce. Z tabulek 3 a 4 je možno také vyčíst, že ve 23 případech bylo globální minimum nalezeno nejpozději v 10 generaci. V jednom případě bylo nutné využít 15 generací a v jednom případě až 22 generací.



Graf 1 Vývoj minimálních hodnot účelové funkce genetického algoritmu pro 30 spuštění.

V grafu 1 je zobrazeno všech 30 měření, kde jednotlivá data představují nejlepší hodnoty účelové funkce pro jednotlivé populace. Tato data jsou uvedena v tabulce 3 a v tabulce 4.

6.1.14 Přechod na další stroj

Vybere se další stroj v řadě a vše se vrací do bodu načtení vstupních dat.

6.2 Řešení problému pomocí diferenciální evoluce

Dalším možným řešením problému je pomocí diferenciální evoluce. Toto řešení problému má hodně společných bodů a výpočtů jako při řešení za pomocí genetického algoritmu.

6.2.1 Nastavení parametrů genetického algoritmu

Jako první je nutné definovat parametry algoritmu diferenciální evoluce algoritmu:

- velikost populace – určuje, kolik jedinců bude v jedné populaci,
- počet populací NP – parametr, který udává maximální počet populací,
- práh křížení CR – udává pravděpodobnost, s jakou se budou jedinci křížit,
- mutační konstanta F - udává pravděpodobnost, s jakou se bude provádět mutace,
- stanovení vyhovující hodnoty účelové funkce – hodnota, podle níž lze určit, zdali je nalezené řešení problému vyhovující.

6.2.2 Vstupní data pro rozvrhování surových plášťů na jednom stroji

Vstupní data jsou stejná jako při řešení problému pomocí genetického algoritmu:

- název (kód) výrobku na stroji – slouží pro přesnou identifikaci výrobku v rozvrhu,
- čas výroby jednoho kusu výrobku na daném stroji,
- počet kusů výrobku odebraných ze systému – stanovení počtu kusů výrobku, které se budou odečítat z aktuálního počtu výrobků za stanovený čas,
- salda výrobků – hodnota udávající stávající stav počtu kusů pro jednotlivé výrobky.

6.2.3 Omezující parametry

Omezující parametry jsou opět stejné jako u řešení pomocí genetických algoritmů:

- maximální počet vyrobených kusů daného výrobku – omezující podmínka, která určuje maximální počet kusů výrobku, který vstupuje do rozvrhu.
- minimální počet kusů výrobku - omezující podmínka, která určuje, zdali je ještě dostatečný počet kusů výrobku.

6.2.4 Vytvoření jedinců první populace

Rovněž vytvoření jedinců lze převzít z genetických algoritmů. Opět je nutné stanovit pořadí rozměrů podle prioritních pravidel, dodržet stejnou délku chromozómu pro všechny jedince v populaci a vypočítat pro všechny rozměry maximální počet vyrobených kusů pro jednu výrobní dávku.

6.2.5 Výpočet účelové funkce

Výpočet účelové funkce lze taktéž beze změn převzít z řešení pomocí genetických algoritmů. Samotný výpočet účelové funkce všech jedinců hledá optimální, v tomto případě minimální hodnoty funkce.

6.2.6 Ukončovací kritéria pro první populaci

Ukončovací kritéria pro první populaci spočívají v nalezení řešení s hodnotou účelové funkce, která je stejná nebo lepší než předem stanovená hodnota účelové funkce. Pokud tato podmínka není splněna, pokračuje se ve vytváření nových jedinců a celých populací. Novou populaci lze vytvořit pomocí výběrů několika jedinců a jejich vzájemné mutace a křížení.

6.2.7 Selekce

Pokud výpočetní algoritmus neskončí v první populaci, je nutné pro další běh algoritmu náhodně vybrat celkem čtyři jedince (strategie DE/rand/1bin) za použití funkce:

```
Function selectGene()
    selectGene = Int(((population_size - 1) - 0 + 1) * Rnd + 0)
End Function
```

Zdrojový kód č. 9 Selekce u diferenciální evoluce.

6.2.8 Mutace

Mutační konstantou a výběrem tří jedinců, z celkového počtu čtyř náhodně vybraných jedinců, lze vytvořit šumový vektor. Nejdříve dva jedinci od sebe odečtou navzájem všechny parametry z chromozómu, které představují počet kusů, které se mají v dané dávce vyrábět. Všechny tyto nové parametry se vynásobí mutační konstantou F. Tímto výpočtem vznikne nový váhový diferenční vektor, u kterého jsou všechny parametry přičteny k parametrům třetího vybraného jedince. Tímto je vypočítán celý šumový vektor.

```
For i = 1 to number_of_items
   pieces(noise_vector) = pieces(vector_X1) + _
   F * (pieces(vector_X2) - pieces(vector_X3))
Next i
```

Zdrojový kód č. 10 Výpočet šumového vektoru (DE).

6.2.9 Křížení

Křížení představuje další krok algoritmu, kdy výsledkem je vytvoření zkušebního vektoru. Tento vektor se určuje za pomocí šumového vektoru a posledního čtvrtého vybraného jedince, který je nazýván jako aktivní jedinec. Je vybrána náhodná hodnota `random_value`, která je porovnávána s prahem křížení CR `CR_value` pro každý parametr vektoru, a jestliže je hodnota `CR_value` větší než hodnota `random_value`, pak na pozici zkušebního parametru připadne hodnota z parametru šumového vektoru, pokud je tomu naopak, pak připadne výsledná hodnota z parametru čtvrtého jedince.

```
If random_value < (CR_value) Then
    pieces_candidate = pieces(noise_vector)
else
    pieces_candidate = pieces(vector_X4)
End if
```

Zdrojový kód č. 11 Křížení u diferenciální evoluce.

U zkušebního parametru je určena hodnota účelové funkce, která je porovnána s hodnotou účelové funkce aktivního jedince. Do nové populace postupuje jedinec, který má nižší hodnotu účelové funkce.

```
If temp_fitness_candidate < temp_fitness_active Then
        population.Item(new) = breed_population.Item(candidate)
        fitness.Item(new) = temp_fitness_candidate
else
        population.Item(new) = breed_population.Item(active)
        fitness.Item(new) = temp_fitness_active
End If
```

Zdrojový kód č. 12 Porovnání jedinců a rozhodnutí, který z nich bude postupovat do nové populace (DE).

6.2.10 Ukončovací kritéria

Ukončovací kritéria jsou rozhodujícím parametrem k ukončení výpočtů na daném stroji. Zde jsou uvedená základní a pro toto řešení dostačující kritéria:

1. pokud je nalezen rozvrh bez prostojů, tedy s minimální hodnotou účelové funkce, ukončí se výpočet algoritmu. Naopak, pokud není dosažena stanovená minimální hodnota účelové funkce, je nutné vytvořit pomocí turnajového výběru, křížení a mutace novou populaci, a opakovat celý výpočet pro novou populaci,

2. stagnace výše hodnoty účelové funkce nejlepšího jedince ve stanoveném počtu po sobě jdoucích generací,

3. jestliže je vyčerpán stanovený počet populací, tak se ukončí výpočet algoritmu.

6.2.11 Vizualizace rozvrhu

Po vytvoření rozvrhu výroby rozměrů pro stroj je rozvrh výroby zobrazen, stejně jako u řešení pomocí genetického algoritmu, přehledně v tabulce (Obr. č. 15). Je zde uveden v prvním sloupci PROSI kód rozměru, dále počet kusů, které se mají vyrábět a čas výroby. Tento rozvrh byl náhodně vygenerován.

5200002833	**80**	13:45:00
5200000390	**175**	14:59:19
5200002079	**270**	17:41:54
5200002833	**410**	21:52:45
5200000390	**215**	4:13:39
5200002079	**158**	7:33:23
5200002833	**160**	10:00:10
5200000390	**220**	12:28:49

Obr. č. 15 Ukázka vizualizace rozvrhu.

Na obrázku č. 16 je uvedena ukázka vizualizace vývoje počtu kusů jednotlivých rozměrů během prvních 10 hodin. Je zde ukázáno zvýraznění, pokud je nedostatek surových plášťů pro daný rozměr, tedy je aktuální počet kusů `quantity(prosi)` je menší nebo roven minimálnímu počtu kusů u rozměru `min(prosi)`.

Tato situace je znázorněna u rozměru 5200002833, kdy v časech mezi 18:45 až 21:45 není dostatečné množství surových plášťů, což je znázorněno červeným pozadím a textem. Minimální počet kusů se vypočítává podle vzorce (25), výpočet aktuálních kusů je podle vzorce (28) a (29).

Rozměr	Minimální počet kusů	13:45	14:45	15:45	16:45	17:45	18:45	19:45	20:45	21:45	22:45
5200002833	63	134	183	151	120	88	57	25	0	0	379
5200000390	37	200	181	338	319	301	282	264	245	227	208
5200002079	28	184	170	156	142	398	384	370	356	342	328

Obr. č. 16 Ukázka vizualizace rozvrhu pro prvních 10 hodin a vývoj počtu kusů v čase, s ukázkou vizualizace nedostatku surových plášťů.

6.2.12 Kontrola výsledků DE rozvrhu pro daný stroj

Pro ověření správnosti výsledků řešení pomocí diferenciální evoluce, byl vybrán stejný stroj a stejné rozměry, jako v případě použití u řešení problému pomocí genetického algoritmu. Potřebné vstupní parametry rozměrů jsou uvedeny v tabulce 5.

Tab. 5 Vstupní parametry surových plášťů – kontrola funkčnosti rozvrhu.

Rozměr PROSI kód	Počet kusů	čas výroby 1 ks	forem 1.sm	forem 2.sm	forem 3.sm	Priorita	lis ks/hod
5200002833	134	0,56	6	6	6	0,044	5,2
5200000390	200	0,56	4	4	4	0,020	4,6
5200002079	184	0,56	3	3	3	0,016	4,7

Nastavení všech parametrů DE, které bylo použito v testu, je zobrazeno v tabulce 6. Nastavení maximálního času salda v hodinách bylo zvoleno na 24 hodin, což je jedna ze vstupních hodnot pro výpočet maximální množství kusů v jedné dávce daného rozměrů podle vzorce (22). Nastavení minimálního času salda v hodinách bylo zvoleno na 4 hodiny, což je také jedna ze vstupních hodnot pro výpočet minimálního množství kusů v jedné dávce daného rozměrů podle vzorce (25).

Tab. 6 Nastavení parametrů diferenciální evoluce – kontrola rozvrhu.

Velikost populace	60
Počet generací NP	25
Práh křížení CR	0,5
Mutační konstanta F	0,8
Maximální čas salda (hod)	24
Minimální čas salda (hod)	4

Cílem bylo, stejně jako v případě řešení pomocí genetických algoritmů, ověřit výpočet rozvrhu, zdali poskytuje podobné výsledky i při více nezávislých spuštěních. Zadání a nastavení bylo ve všech případech stejné, jen výpočet se spustil celkem třicetkrát po sobě, a vždy se stejnými výchozími podmínkami. V tabulkách 7 a 8 řádky prezentují číslo generace, a sloupce představují pořadové číslo spuštění. Uvedené hodnoty v tabulce představují hodnotu účelové funkce nejlepšího jedince v populaci. Nejlepší hodnota je prezentována nulou. Vyšší velikost hodnoty účelové funkce znamená rostoucí chybu v rozvrhu. Pro účely kontroly výsledků rozvrhu nebylo využito ukončovací kritérium, které ukončí výpočet po nalezení nulové hodnoty účelové funkce.

Z tabulky 7 lze vyčíst, jak se pohybovala velikost populace u prvních patnácti spuštění a dalších 15 spuštění je zobrazeno v tabulce 8. Z obou těchto tabulek je patrno, že ve všech případech bylo nalezeno globální minimum funkce, které reprezentuje nulové prostojové hodiny na nedostatek surových plášťů pro lisovnu.

Tab. 7 Kontrolní výpočty diferenciální evoluce pro prvních 15 výpočtů.

Počet generací \ Počet měření	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	75	290	130	45	0	15	15	120	180	140	160	210	215	160	150
2	75	280	130	30	0	15	15	120	180	140	160	210	180	90	150
3	75	280	130	30	0	15	15	120	105	140	65	180	120	90	150
4	75	120	35	30	0	15	0	120	105	40	60	180	120	60	150
5	75	120	35	30	0	15	0	75	105	40	60	120	90	60	125
6	75	120	35	30	0	15	0	60	105	40	60	80	80	60	125
7	40	120	35	30	0	15	0	60	40	40	60	80	80	60	125
8	40	75	35	30	0	15	0	50	40	40	60	15	80	0	20
9	40	0	35	30	0	15	0	20	40	0	60	15	20	0	20
10	40	0	35	30	0	0	0	20	0	0	60	15	20	0	0
11	0	0	35	30	0	0	0	20	0	0	35	0	20	0	0
12	0	0	35	30	0	0	0	20	0	0	35	0	20	0	0
13	0	0	35	0	0	0	0	20	0	0	35	0	0	0	0
14	0	0	35	0	0	0	0	20	0	0	35	0	0	0	0
15	0	0	35	0	0	0	0	20	0	0	15	0	0	0	0
16	0	0	35	0	0	0	0	0	0	0	15	0	0	0	0
17	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

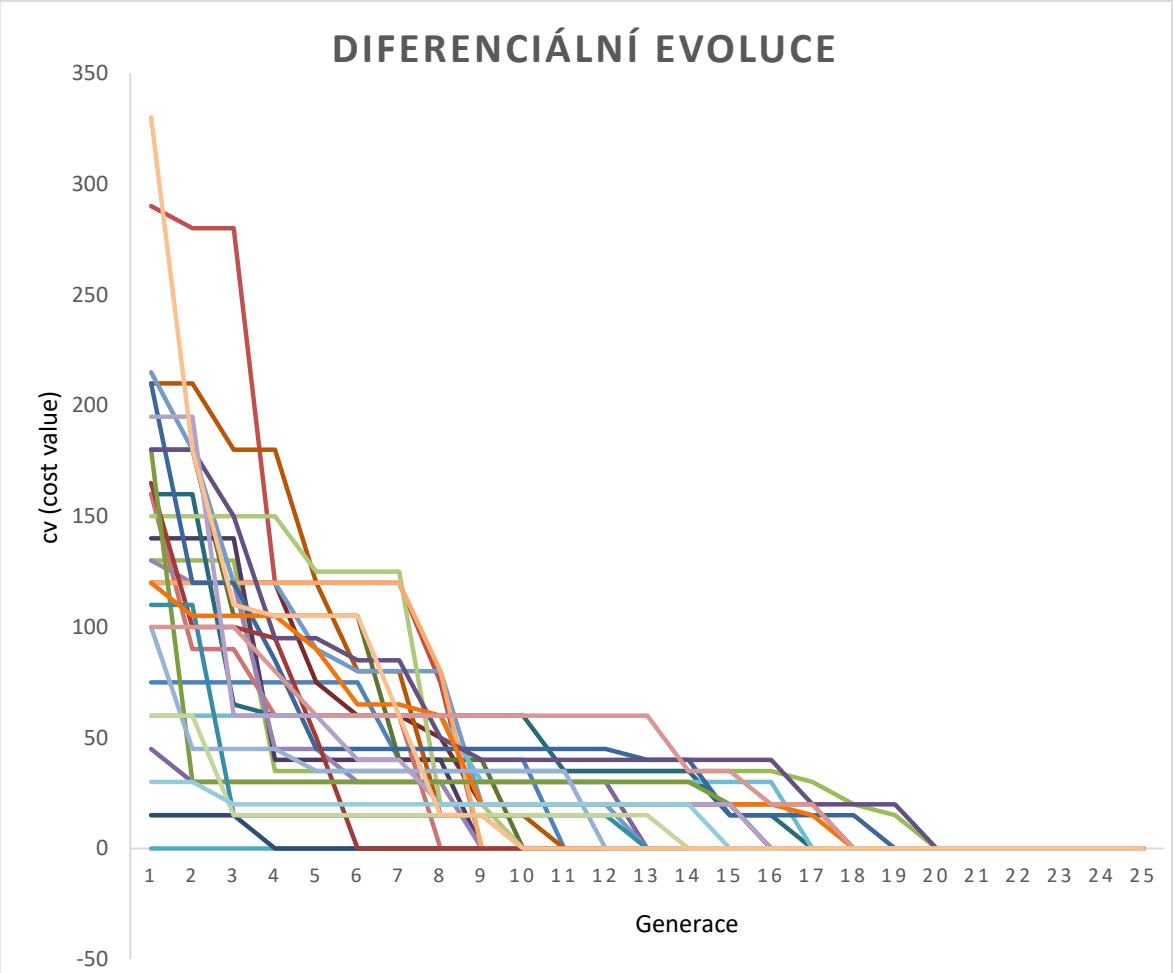
Z tabulky 7 lze určit, že hledané globální minimum bylo nalezeno ve všech 15 případech. Z toho v jednom případě ihned v první generaci. V 8 případech bylo toto globální minimum nalezeno nejpozději v 10. generaci. Ve třech případech bylo nutné využít minimálně 15 generací, a v jednom případě až 20 generací. Ve všech prvních generacích se hodnota účelové funkce pohybovala od ideální nuly až po hodnotu 290 a celkový průměr byl 127. Průměrná hodnota účelové funkce v desáté generaci byla již 14,7, a hodnoty se pohybovaly se mezi nulou a 60. Při všech patnácti spuštěních byla, od dvacáté generace, hodnota účelové funkce rovna nule, tedy byl nalezen rozvrh výroby surových plášťů pro jeden stroj bez prostojových hodin na lisovně.

Tab. 8 Kontrolní výpočty diferenciální evoluce pro dalších 15 výpočtů.

Počet generací	Počet měření														
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	130	60	120	210	165	180	180	110	120	100	100	60	195	30	330
2	120	60	120	120	100	30	180	110	105	45	100	60	195	30	180
3	120	60	120	120	100	30	150	15	105	45	100	15	60	20	110
4	45	60	120	85	95	30	95	15	105	45	80	15	60	20	105
5	45	60	120	45	50	30	95	15	90	35	60	15	60	20	105
6	30	60	120	45	0	30	85	15	65	35	60	15	40	20	105
7	30	60	120	45	0	30	85	15	65	35	60	15	40	20	60
8	30	60	80	45	0	30	50	15	60	35	60	15	20	20	15
9	0	30	0	45	0	30	40	15	20	35	60	15	20	20	15
10	0	30	0	45	0	30	40	15	20	35	60	15	20	20	0
11	0	30	0	45	0	30	40	15	20	35	60	15	20	20	0
12	0	30	0	45	0	30	40	15	20	0	60	15	20	20	0
13	0	30	0	40	0	30	40	0	20	0	60	15	20	20	0
14	0	30	0	40	0	30	40	0	20	0	35	0	20	20	0
15	0	30	0	15	0	20	40	0	20	0	35	0	20	0	0
16	0	30	0	15	0	20	40	0	20	0	20	0	0	0	0
17	0	0	0	15	0	20	20	0	15	0	20	0	0	0	0
18	0	0	0	15	0	0	20	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

V případě dalších 15 výpočtů, uvedených v tabulce 8, lze říci, že opět bylo hledané globální minimum nalezeno ve všech patnácti případech. Ve čtyřech případech bylo toto globální minimum nalezeno nejpozději v desáté generaci. V sedmi případech bylo nutné využít minimálně 15 generací, a v jednom případě až devatenáctou generaci. Ve všech prvních generacích se hodnota účelové funkce pohybovala od hodnoty 30 až po hodnotu 330, a celkový průměr byl 139.3. Průměrná hodnota účelové funkce v desáté generaci byla 22, a pohybovala se mezi nulou a 60.

Ve všech třiceti měřeních (tabulka 7 a 8) bylo nalezeno ve 25. generaci globální minimum účelové funkce.

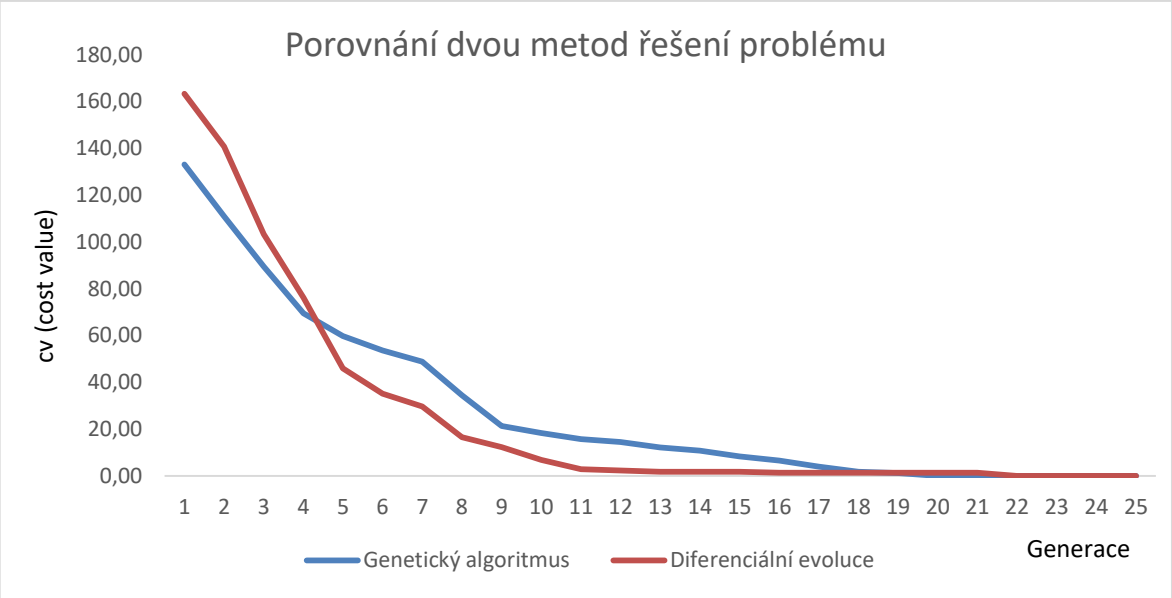


Graf 2 Vývoj hodnot účelové funkce diferenciální evoluce pro 30 výpočtů.

V grafu 2 je zobrazeno všech 30 měření, přičemž jednotlivá data představují nejlepší hodnoty účelové funkce pro jednotlivé populace. Tato data jsou také uvedena v tabulce 7 a v tabulce 8.

6.3 Porovnání řešení problému pomocí GA a DE

Z předchozích kontrolních výpočtů obou metod řešení problému rozvrhování, byl stanoven, zvlášť pro každou metodu, průměr nejlepších hodnot pro jednotlivé generace a následně vzájemně porovnán. Graf č. 3 ukazuje, že v případě řešení problému pomocí diferenciální evoluce je možné říci, že jsou dosaženy rychleji velmi nízké hodnoty účelové funkce, ale globální minimum naleznou obě řešení ve velmi podobném počtu populací.



Graf 3 Porovnání průměrných hodnot nejlepších jedinců v generaci u genetických algoritmů a diferenciální evoluce.

Z grafu 3 lze vyvodit, že obě řešení, tedy pomocí genetického algoritmu i diferenciální evoluce, mají podobné výsledky. To vede k závěru, že obě metody si jsou velmi podobné s ohledem na výsledky řešení. Hodnota výpočetního času, kdy byl prováděn výpočet, byl rovněž u obou metod podobný, ale tyto hodnoty nebyly předmětem podrobného zkoumání.

Původní myšlenkou bylo také zobrazit v grafu i porovnání obou metod řešení v jejich nejlepších, a tím pádem i nejrychlejších případech, kdy bylo nalezeno globální minimum, ale v případě genetického algoritmu bylo toto minimum nalezeno již ve třetí generaci a u řešení s diferenciální evolucí to bylo dokonce ihned v první generaci. Tím by reprezentace výsledků pomocí grafu nebyla nejvhodnější.

Tabulka 9 představuje statistické údaje hodnot účelové funkce u poslední 25. generace (z tabulek 3, 4, 7 a 8), pro obě porovnávaná řešení problému. Jelikož ve všech případech byla hodnota účelové funkce rovna nule, tak i minimální a maximální hodnota účelové funkce pro poslední 25. generaci se rovná nule.

Tab. 9 Statické údaje hodnot účelové funkce poslední generace pro GA a DE.

hodnoty účelové funkce	GA	DE
min	0	0
max	0	0

6.4 Realizace aplikace

Jako součást této práce byly vytvořeny dvě aplikace pro software Microsoft Excel, který je používán pro plánování v Continental Barum s.r.o. (divize CoBa). Aplikace jsou napsány v programovacím jazyce Visual Basic for Applications, který je používán jako součást balíku Microsoft Office, a také využívá pro řazení dat do dynamického pole třídu .NET Frameworku (System.Collections.Arraylist, knihovna mscorlib). Generování rozvrhu je v jedné aplikaci vytvořeno za pomocí genetických algoritmů, a v druhé aplikaci pomocí diferenciální evoluce. Obě aplikace umožňují vytvořit rozvrh pro všech 39 modulů, kdy u každého modulu zvlášť je aplikován výpočet pomocí evolučních výpočetních technik pro jeden stroj.

Výpočet operativního rozvrhu je možné rozdělit na dvě části. První část představuje přípravu dat pro pozdější výpočet a samotný výpočet pomocí evolučních výpočetních technik.

6.4.1 Příprava dat

Aby bylo možné vytvářet rozvrh výroby pro jeden stroj, je nutné nejdříve načíst potřebná data shodná pro obě aplikace. Tato data představují:

- plán rozměrů na stroji pro následujících 24 hodin. Tento plán se bere od hlavního plánovače,
- přidělení času výroby jednoho kusu výrobku z plánu,
- plán počtu forem jednotlivých rozměrů během následujících 24 hodin a jejich časy na vylisování jednoho kusu výrobku,
- aktuální počet surových plášťů pro všechny rozměry

Poté se vypočítají všechna přípravná data pro všechny rozměry:

- podle vzorce 22 se určí maximum kusů na jednu dávku výroby rozměru,
- podle vzorce 23 se vytvoří priorita výroby rozměrů na stroji,
- podle vzorce 24 se vypočítá počet vylisovaných kusů za hodinu,
- podle vzorce 25 se určí minimální počet kusů.

Tab. 10 Ukázka připravených dat pro výpočet pomocí EVT.

Stroj	Rozměr PROSI kód	Počet kusů	Forem aktuální směna	Forem směna+1	Forem směna+2	Minimum kusů	Maximum kusů	Vylisovaných ks/hod
B5	5200002833	134	6	6	6	126	755	31
B5	5200000390	200	4	4	4	74	445	19
B5	5200002079	184	3	3	3	56	337	14

Tato příprava probíhá najednou pro všech 39 modulů (strojů). Takto je vytvořen seznam ve stejném formátu, jako je uveden v tabulce 10. Součástí přípravy dat je i zobrazení pro jednotlivé stroje, v jakém mají průměrné aktuální časové saldo surových plášťů ze všech rozměrů (Obr č. 17). Jedná se v podstatě o zobrazení, jakou má daný stroj výchozí pozici pro splnění svého rozvrhu. Čím vyšší číslo je uvedeno, tím je na tom stroj lépe.

	Stroj
8,5	E1
7,0	E2
5,2	E3
6,5	E4
6,2	E5
8,7	G1
10,7	G2
11,5	G3
3,6	G4
6,3	G5
8,1	G6
7,0	H1
8,0	H2
7,9	H3
4,0	H4
12,5	H5
11,1	H6
13,8	H7

Obr. č. 17 Ukázka aktuálního časového salda pro jednotlivé stroje.

6.4.2 Výpočet a vytvoření rozvrhu

Uživatel má u obou aplikací možnost si vybrat, zdali chce vytvořit rozvrh pro všechny moduly automaticky, nebo si vybere možnost manuálně nakonfigurovat stroj a poté spustit výpočet pouze pro tento stroj. V případě, že uživatel zvolí možnost individuálního nastavení, lze do tabulky zadat jednotlivé rozměry a k nim buď ponechat vygenerovaná data z přípravy, nebo je možnost si plně přizpůsobit všechna data podle vlastního uvážení (Obr. č. 18).

Stroj:	A2
Kapacita:	1600

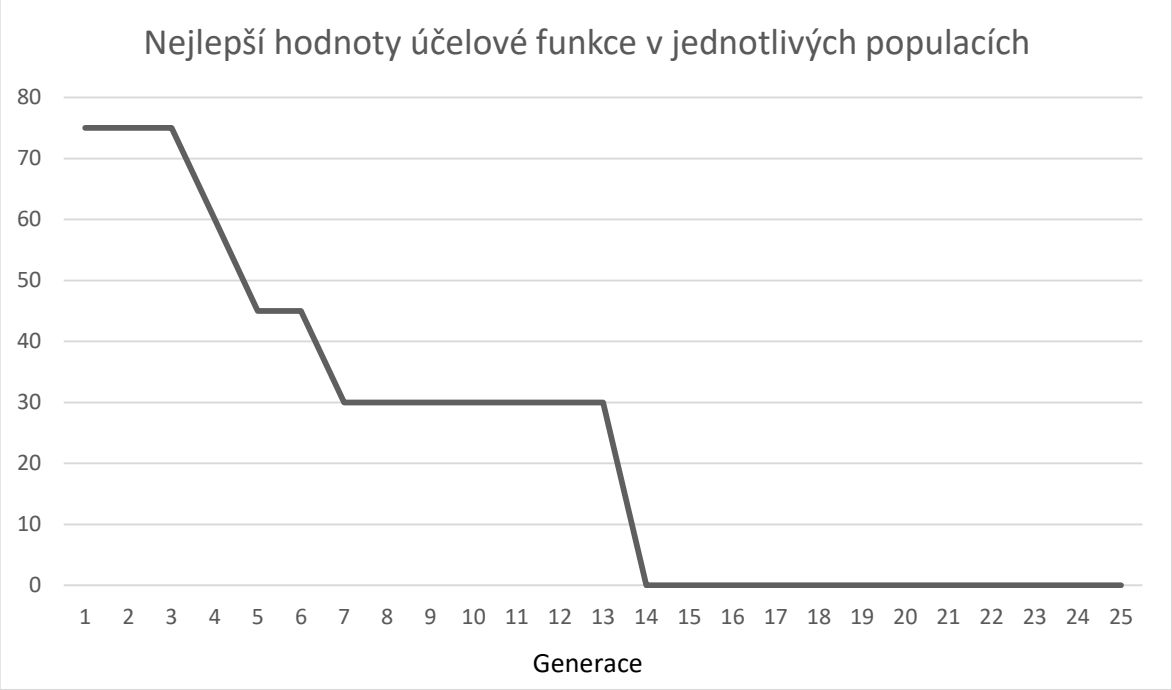
	Prosi		Saldo	Forem sm	Forem sm+1	Forem sm+2	Priorita	lis ks/hod
1.	6583	5200006583	187	7	7	7	0,0374	4,8
2.	6617	5200006617	63	2	2	2	0,0317	4,5
3.	6584	5200006584	158	4	4	4	0,0253	4,6
4.	600	5200000600	117	2	2	2	0,0171	4,4

Obr. č. 18. Individuální příprava pro jeden stroj.

Po tomto uspořádání, výběru stroje a jeho kapacity, je možné spustit samotný výpočet. Následující postup je společný i při individuálním přístupu tvorby rozvrhu, tak i při plně automatickém, kdy samotný výpočet rozvrhu je podrobně popsán, v případě první aplikace za pomocí genetického algoritmu, v kapitole 6.1, a pro výpočet v druhé aplikaci, za pomocí diferenciální evoluce, v kapitole 6.2.

Společnou částí pro obě evoluční výpočetní techniky je také vizuální výstup pro jednotlivé stroje, obdobně jako je tomu např. na Obr. č. 15 (str. 50), na kterých je možné i po vygenerování rozvrhu ručně upravovat počet kusů pro jednotlivé rozměry a dávky. Po této volitelné úpravě se ihned provede výpočet, a zobrazí se průběh salda u všech rozměrů i s novým výpočtem prostojů způsobených změnou jednotlivých dávek.

V případě individuálního zadání dat pro jeden stroj lze po výpočtu a zobrazení rozvrhu také vidět vývoj hodnot účelové funkce, stejně jako v Grafu 4.



Graf 4 Vývoj hodnot účelové funkce u individuálního výpočtu.

Graf č. 4 čerpá z nejlepších hodnot účelové funkce u jednotlivých populací, které jsou pro každou populaci vytvořeny a následně vybrány jako nejlepší řešení problému v dané populaci. Ukázka jedinců seřazených podle nejlepších (minimálních) hodnot účelové funkce u poslední 25. populace je v tabulce 11.

Tab. 11 Ukázka prvních 30 jedinců poslední generace, seřazených podle CV.

Jedinec v populaci	Hodnota účelové funkce
3000801604000601601200803 80	0
3000801604000601601200803 80	0
3000801604000601601200803 80	0
3000801604000601601200803 80	0
3000801604000601601200803 80	0
3000801604000602601201604 20	30
3000801604000602601203404 40	30
3000801604000602601203404 40	30
3000801604000602601203404 40	30
3000801604000602601200803 80	30
3000801604000602601203400 80	30
3000801604000602601203404 40	30
3000801604000602601203403 80	30
3000801604000602601200803 80	30
3000801604000602601203403 80	30
3000801604000602601203000 80	30
3000801604000602601203404 40	30
3000801604000602601200803 80	30
3000801604000602601203403 80	30
3000801604000602601201801 20	30
3000801604000602601203403 80	30
2800801604000803201603404 40	30
3000801604000602601200803 80	30
3000801604000602601203404 40	30
3000801604000602601203404 40	30
2800801604000602601203404 40	30
3000801604000602601200803 80	30
3000801604000602601203404 40	30
3000801604000602601203404 40	30
3000801604000602601200803 80	30

I následující souhrn prostojových hodin za 24 hodin (Obr. č. 19) je společný pro obě aplikace. Jedná se o zobrazení, kolik který stroj, podle pro něj vygenerovaného rozvrhu, by měl mít prostojových hodin na lisovně za následujících 24 hodin, respektive jeho rozměry (`downTime(prosi)`, str. 39). Za pomocí tohoto souhrnu je vhodné zvážit, jestli nelze stroji, na kterém jsou uvedeny prostojové hodiny, nějakým způsobem pomoci (přesunem rozměrů, výpomoc na jiném stroji, atd.).

Prostojové hodiny mohou také vzniknout poruchou, nebo plánovanou opravou, přičemž ne všechny stroje jsou identické. Na některých strojích se vyrábí složitější výrobky, které jsou ale obvykle také nejvíce požadované od zákazníků. Tyto rozměry většinou nelze, zejména kvůli své specifikaci, vyrábět na jiných strojích.

A1	A2	A3	A4	A5	A6	A7
0	0	0	0	0	0	10

B1	B2	B3	B4	B5	B6	B7
0	6	0	0	0	0	0

C1	C2	C3	C4	C5	C6	C7
34	27	1	0	0	0	2

E1	E2	E3	E4	E5
2	3	27	0	0

G1	G2	G3	G4	G5	G6
0	0	2	26	13	0

H1	H2	H3	H4	H5	H6	H7
0	6	4	47	0	0	0

Celkem 210 hodin prostojů

Obr. č. 19 Ukázka předpokládaných prostojových hodin za 24 hodin.

Z obrázku č. 19 lze vyčíst, že stroje C1, C2, E3, G4 a H4 jsou nejvíce ohroženy tím, že nestihnou v čas vyrobit dostatečné množství surových plášťů pro svého zákazníka, tedy lisovnu. Konkrétně u těchto strojů jsou sebemenší poruchy či zdržení velmi citelná, a ihned se projeví v prostojích. Bohužel, vzhledem k jejich vyráběnému sortimentu je nelze jednoduše nahradit jinými stroji.

Jako hlavní výstup obou aplikací, při volbě vytvoření rozvrhu pro všechny moduly, je rozvrh všech rozměrů na 24 hodin pro všech 39 modulů (tabulka 12, str. 62). Rozvrh všech rozměrů může být poté seřazen podle abecedního pořadí názvů strojů nebo podle času, což lze využít i při rozvrhování výroby polotovarů.

Takto stanovený plán na 24 hodin je možné poté načítat do souboru, který doposud sloužil pro ruční rozvrhování výroby surových plášťů na jednu směnu. Toto načítání plánu na 24 hodin již není součástí, v této práci vytvořených aplikací, ale je nově plně integrované do stávajícího souboru rozvrhování výroby. Tento soubor umožňuje tisk rozvrhu pro jednotlivé stroje v požadovaném formátu.

7 POROVNÁNÍ DOSAVADNÍHO A NOVÉHO ROZVRHOVÁNÍ

Rozvrhování výroby surových plášťů, na konfekci v divizi CoBa, doposud probíhalo operativně. Celá výroba od polotovarů, přes výrobu surových pláštů, až po vylisování plášťů využívalo systém Kanban, který je úzce spojený se systémem štíhlé výroby Just In Time (JIT). Pro daný stroj byl, stejně jako nyní, vytvořen plán rozměrů, které se mají na tomto stroji vyrábět příštích 24 hodin. Rozvrhování rozměrů a jejich počet kusů ve výrobní dávce závisel na pověřených pracovnících, koordinátorech výroby, kteří rozepisovali výrobu na další směnu podle počtu kanbanovaných karet pro 39 modulů. Tento systém fungoval do té doby, dokud se počet plánovaných rozměrů na 24 hodin pohyboval do 140 s celkovým počtem 50 000 vyrobených kusů za 24 hodin. S překročením hranice 140 plánovaných výrobků nastal problém s organizací výroby přípravných materiálů a samotného rozvrhování výroby surových plášťů. A při aktuálním počtu 160 výrobků se systém rozvrhování pomocí kanban karet stal méně přesným, než by bylo žádoucí. Bylo velmi obtížné se rozhodnout, kterému rozměru dát přednost, aby byla plně pokryta výroba na konfekci a zároveň, aby byl co nejlépe uspokojen zákazník, tedy lisovna. Počet koordinátorů výroby se tedy navýšil na tři lidi na směnu. Tito pracovníci měli jednak na starost rozvrhování výroby na konci směny, kdy připravovali rozvrh na následující směnu, a také kontrolu kvality výroby během probíhající směny. Takto nastavený systém chvíli fungoval, ale projevil se další problém. Jelikož se rozvrh výroby rozepisoval pouze na následujících 8 hodin, příprava materiálů neměla přesnou informaci, jaký výrobek se bude po těchto 8 hodinách vyrábět, a vcelku logicky vyráběli podle kanban karet. Jenže se stále častěji stávalo, že vyrobené polotovary byly potřeba např. až v druhé půli směny, a ty rozměry, které bylo třeba vyrábět pro lisovnu na začátku směny, neměly nachystané polotovary. Tato situace se řešila tak, že na začátku každé směny příchozí koordinátoři přizpůsobili rozvrh výroby materiálům, které byly k dispozici, ale tím byla pouze částečně zajištěna potřebná výroba rozměrů pro lisovnu. Veškeré pokusy o regulaci počtu kanban karet, bohužel, nevedl k uspokojivému výsledku. Proto vzešel požadavek na vytvoření rozvrhu výroby, který by pokrýval delší časový úsek, než dosavadních 8 hodin.

V rámci této práce byly vytvořeny dvě aplikace, řešící problém rozvrhování výroby pomocí výpočetních evolučních technik. V jednom případě pomocí genetických algoritmů a v druhém případě pomocí diferenciální evoluce.

Do praxe byla uvedena varianta řešení problému rozvrhování surových plášťů pomocí genetického algoritmu, zejména z důvodu rychlosti výpočtu a stabilních výsledků, jejichž kontrola je popsána v kapitole 6.2.12.

Uživatel aplikace, v tomto případě dispečer výroby, spustí výpočet rozvrhu výroby surových plášťů na 24 hodin pro všechny stroje, a poté si touto aplikací vytvořený rozvrh výroby přebírají jednotlivá oddělení. Tento rozvrh je zároveň plně přizpůsoben požadavkům lisovny a také respektuje možnosti výroby surových plášťů na konfekci.

Tab. 12 Ukázka plánu na 24 hodin, seřazeno podle stroje.

stroj	Rozměr	kusy	čas
B10	5200004020	360	20:15:00
B10	5200006583	200	1:39:00
B10	5200004318	180	4:39:00
B10	5200006564	80	7:21:00
B10	5200004020	360	8:33:00
B10	5200006583	200	13:57:00
B10	5200004318	180	16:57:00
B10	5200002497	140	19:39:00
B10	5200004020	400	21:45:00

Hlavní výhodou takto vytvořeného rozvrhu je jeho rozsah, který je na 24 hodin. Každé oddělení výroby si vezme z tohoto plánu jasně definovaný časový úsek, který může podle svých možností navýšit. Tímto je zaručeno, že všichni pracují na stejných rozměrech tak, aby byly všechny polotovary včas a v potřebném množství na konfekci, která s předstihem vyrábí pro lisovnu.

Například pro rozpis výroby na konfekci je, v její speciální aplikaci, rozvrh na 24 hodin převzat a seřazen podle strojů, stejně jako v tabulce 12, kdy na jednotlivou směnu se pro konkrétní stroj vyberou pouze ty rozměry, které jsou určené, z časového hlediska, na danou směnu.

Naopak pro přípravu materiálů se plán na 24 hodin seřadí podle času, bez ohledu na stroje na konfekci, stejně jako je tomu u ukázky prvních desíti rozměrů seřazených podle času v tabulce 13. Poté je možné podle aktuálního salda polotovarů určit, které polotovary je nutné vyrábět, a které je možné odložit na později.

Tab. 13 Ukázka plánu na 24 hodin, seřazeno podle času.

stroj	Rozměr	kusy	čas
A10	5200006157	140	20:15:00
A20	5200003462	80	20:15:00
A30	5200006565	80	20:15:00
A40	5200005880	60	20:15:00
A50	5200006099	400	20:15:00
A60	5200005867	140	20:15:00
A70	5200003286	120	20:15:00
B10	5200004020	360	20:15:00
B20	5200006214	260	20:15:00
B30	5200002174	100	20:15:00

Příklad takto vytvořeného plánu výroby běhounů na vytlačovací lince číslo 6 je v tabulce 14. Rozvrh pro tuto linku je vytvářen na následující dvě směny, čímž je zajištěno, že se budou vyrábět právě ty rozměry, které jsou potřeba pro aktuální směnu, a zároveň je možné lehce určit, které rozměry se budou vyrábět na konfekci v následující směně a i s přesným pořadím. Z tabulky 14 lze také vyvodit, že tato vytlačovací linka má vyrobeno dostatek běhounů pro převážnou část přidělených rozměrů, neboť je potřeba vyrábět v příštích hodinách pouze 13 rozměrů z celkového počtu přidělených 48 rozměrů v daný den.

Tab. 14 Ukázka rozvrhu výroby pro vytlačovací linku 6.

Rozměr	čas	Směs	VL	Vyrábět kusů
5200006077	22:20:00	T2128	6	409
5200003563	22:21:00	T2190	6	330
5200006583	1:39:00	T2190	6	210
5200005789	2:39:00	T4053	6	70
5200006379	5:34:04	T11200	6	94
5200005867	6:43:22	T1229	6	160
5200006744	7:07:48	T11347	6	130
5200004020	8:33:00	T11200	6	169
5200005721	9:45:00	T12301	6	120
5200003286	11:15:00	T6285	6	444
5200006734	11:33:00	T6220	6	260
5200006099	12:15:00	T2107	6	431
5200006338	12:54:32	T2190	6	80

Samotné srovnání obou přístupů rozvrhování, jak původního operativního za pomocí systému Kanban, tak nového rozvrhování pomocí genetického algoritmu, je vcelku obtížné, neboť rozvrh, který vygeneruje genetický algoritmus, nelze přesně zatížit poruchami a jinými neplánovanými nedostatky, jako v případě již proběhnutého rozvrhování. Pokud byl přesto tento rozvrh na 24 hodin, pro všechny moduly, generován pomocí genetického algoritmu, a za stejných výchozích podmínek, jako při rozvrhování výroby koordinátory, a bylo maximálně přihlédnuto k omezením v daný den, tak vykazoval tento model rozvrhu o dvě třetiny menší prostojové hodiny za 24 hodin, než koordinátory vytvořený rozvrh.

Pro plné ověření funkčnosti vytvořené aplikace, která vytváří pomocí genetického algoritmu rozvrh surových plášťů na 24 hodin, bylo možné jedině ostré nasazení do výroby. Předpokládalo se, že dojde v prvních 16. hodinách k celkovému poklesu výroby, než se veškeré polotovary a rozvržení rozměrů na strojích ustálí, ale samotné nasazení nového systému proběhlo překvapivě snadno. Nejenom, že nedošlo k poklesu výroby v očekávaném období, ale postupně se zvedala výroba a prostojové hodiny na lisovně se snížily. Díky zvýšeným poruchám sice nedošlo k dvoutřetinovému poklesu, ale pouze přibližně polovičnímu snížení prostojových hodin za sledované období 7 dní. V tabulce 15 jsou uvedeny vykázané prostojové hodiny 7 dní před zavedením nového rozvrhování a v tabulce 16 prvních 7 dní nového způsobu rozvrhování. Opět je nutné upozornit, že data nelze jednoznačně porovnat, protože podmínky nejsou, ani nemohou být shodné.

Tab. 15 Počet prostojových hodin za 24 hodin, 7 dní před spuštěním rozvrhování GA.

Pořadové číslo dne	1	2	3	4	5	6	7
Počet prostojových hodin	1030	989	1350	1160	933	820	885

Tab. 16 Počet prostojových hodin za 24 hodin, 7 dní po spuštění rozvrhování GA.

Pořadové číslo dne	1	2	3	4	5	6	7
Počet prostojových hodin	580	520	559	665	538	620	488

Jednoznačným a velmi důležitým přínosem nového rozvrhování je fakt, že i přes velký počet rozměrů (více jak 160 rozměrů a denní výrobní kapacita 50 000 kusů), je příprava materiálů schopna dodávat materiály včas a v potřebném množství. Tím je stanovena dobrá výchozí pozice pro výrobu surových plášťů na konfekci, která může, podle vytvořeného rozvrhu na 24 hodin, vyrábět včas surové pláště rozměrů, které jsou požadovány od jejich zákazníka (lisovny).

ZÁVĚR

Tato diplomová práce je věnována rozvrhování surových plášťů pomocí evolučních výpočetních technik na jednom stroji. Cílem bylo vytvořit nejlepší možný rozvrh výroby surových plášťů na konfekci ve firmě Continental Barum s.r.o., s přihlédnutím na měnící se požadavky potřeby plášťů u zákazníka konfekce, tedy lisovny.

V teoretické části byla provedena analýza problematiky rozvrhování, která představila základní typy úloh rozvrhování, přiblížila čtenáři pojem rozvrh a jeho nejčastější kritéria, a také přehled metod pro plánování a rozvrhování výroby.

Dále podrobně seznamuje se dvěma optimalizačními evolučními algoritmy, které jsou založené na myšlence napodobení vývoje a učení uplatňující se v přírodě. Jako první je popsán genetický algoritmus, u nějž je podrobně popsán princip činnosti algoritmu, vytváření jedinců v populaci, jejich vzájemné křížení a následná mutace. Druhým v pořadí je popis diferenciální evoluce. Jsou zmíněny řídící parametry, popis činnosti a vybrané strategie diferenciální evoluce.

V praktické části je navrhnut výpočet účelové funkce, pomocí níž lze určit, která řešení problému rozvrhování výroby na jednom stroji jsou nejvhodnější. Hlavním úkolem je hledání globálního minima funkce a tím stanovit nejlepší výsledek pro daný problém.

Účelová funkce je poté využita při praktické aplikaci problému, kdy byly vytvořeny dvě aplikace. Jedna se zabývá problémem rozvrhování výroby pomocí genetického algoritmu a druhá za pomocí diferenciální evoluce. Podrobně je popsán postup při tvorbě obou aplikací, jejich následné vyhodnocení a vzájemné porovnání. Obě aplikace prokázaly schopnost nalézt dostatečně kvalitní řešení s optimálními výsledky pro stanovený problém rozvrhování.

Na základě dobrých výsledků při řešení problému byla vybrána aplikace využívající principu genetického algoritmu. Tato aplikace byla nasazena do výroby a poté následně porovnána s dosavadním řešením, které spočívalo v rozvrhování výroby pomocí Kanban systému. Nově navržené řešení se po týdnu zkušebního provozu jeví jako celkově vhodnější než stávající. Předností nového systému je dodržování přesného pořadí a množství vyrobených rozměrů s ohledem na potřeby maximálního využití strojního zařízení na konfekci a včasného dodání surových plášťů pro lisovnu. Dalším přínosem je také znalost přesného času výroby jednotlivých rozměrů a jejich výrobních dávek v následujících 24. hodinách, čehož lze úspěšně využít i při přípravě potřebných polotovarů.

SEZNAM POUŽITÉ LITERATURY

[1] ZELINKA I., OPLATKOVÁ Z., ŠEĎA M., OŠMERA P., VČELAŘ F. *Evoluční výpočetní techniky: principy a aplikace*. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3.

[2] TOMEK, Gustav a Věra VÁVROVÁ. *Integrované řízení výroby: od operativního řízení výroby k dodavatelskému řetězci*. Praha: Grada, 2014. Expert (Grada). ISBN 978-80-247-4486-5.

[3] TRENZ, Oldřich. *Umělá inteligence I* [online]. Brno, 2009 [cit. 2017-05-07]. Dostupné z: http://is.mendelu.cz/eknihovna/opory/index.pl?opora=2068

[4] BAKER, Kenneth R.; TRIETSCH, Dan. *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[5] MAŘÍK, Radek. *Rozvrhování* [online]. Praha, 2013 [cit. 2017-05-07]. Dostupné z: https://cw.fel.cvut.cz/wiki/_media/courses/a3m33ui/prednasky/ui09_rozvrhovani.pdf

[6] RUDOVÁ, Hana. *Rozvrhování* [online]. Brno, 2015 [cit. 2017-05-07]. Dostupné z: https://www.fi.muni.cz/~hanka/rozvrhovani_2015/prusvitky/all_bw.pdf

[7] *Algoritmizace* [online]. Praha, 2009 [cit. 2017-05-07]. Dostupné z: https://cw.fel.cvut.cz/wiki/_media/courses/y36alg/2009_sumperk_p10.pdf

[8] TESAŘ, Karel a Martin MAREŠ. *Složitost* [online]. Praha, 2012 [cit. 2017-05-07]. Dostupné z: https://ksp.mff.cuni.cz/kucharky/slozitost/

[9] DEMLOVA, Marie. *Teorie algoritmů* [online]. Praha, 2016 [cit. 2017-05-07]. Dostupné z: http://math.feld.cvut.cz/demlova/teaching/tal/p-tal5dohr.pdf

[10] MAJER, Petr. Úlohy rozvrhování lze formulovat jako úlohy celočíselného programování a řešit pomocí optimalizačních metod. [online]. Brno, 2003 [cit. 2017-05-07]. Dostupné z: http://www.vutium.vutbr.cz/tituly/pdf/ukazka/80-214-2530-X.pdf

[11] SAWA, Zdeněk. *Algoritmy a výpočetní složitost* [online]. Ostrava, 2005 [cit. 2017-05-07]. Dostupné z: http://wiki.cs.vsb.cz/images/a/af/Algoritmy.pdf

[12] MATOUŠEK, Václav a Petr HELLER. *Umělá inteligence a rozpoznávání* [online]. Plzeň, 2015 [cit. 2017-05-07]. Dostupné z: http://www.kiv.zcu.cz/studies/predmety/uir/predn/P3/FThema3.pdf

[13] HAUPT, Randy L. a S. E. HAUPT. *Practical genetic algorithms*. 2nd ed. Hoboken, N.J.: John Wiley, c2004. ISBN 0-471-45565-2.

[14] HYNEK, Josef. *Genetické algoritmy a genetické programování*. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.

[15] O Teorii omezení. *Goldratt CZ* [online]. Praha, 2015 [cit. 2017-05-07]. Dostupné z: http://www.goldratt.cz/teorie-omezeni/o-teorii-omezeni

[16] KOBLASA, František. *Uplatnění optimalizačních metod v dílenském rozvrhování výrobních zakázek*. In Výrobní systémy dnes a zítra 2008. 2009.

[17] BÍLEK, Ivo. *Aproximativní a heuristické metody řešení NP-těžkých problémů*.VUT v Brně FSI. 2007

[18] RAK, Martin. *Historie výroby pneumatik na Zlínsku* [online]. Olomouc, 2013 [cit. 2017-05-07]. Dostupné z: https://theses.cz/id/a3vjye/Diplomov_prce.pdf

[19] GODFREY C. ONWUBOLU a B.V. BABU. *New optimization techniques in engineering*. Berlin: Springer, 2004. ISBN 9783642057670.

[20] ZELINKA, Ivan. *Umělá inteligence: v problémech globální optimalizace*. Praha: BEN - technická literatura, 2002. ISBN 80-7300-069-5.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

.NET	Platforma pro vývoj software od společnosti Microsoft.
APS	Advanced Planning & Scheduling.
C1	Označení stroje na konfekci firmy Continental Barum s.r.o.
CoBa	Continental Barum s.r.o.
CR	Práh křížení.
D	Dimenze problému.
DBR	Drum Buffer Rope.
DE	Diferenciální evoluce.
E3	Označení stroje na konfekci firmy Continental Barum s.r.o.
EA	Evoluční algoritmy.
EVT	Evoluční výpočetní techniky.
FIFO	First In, First Out.
GA	Genetický algoritmus.
H4	Označení stroje na konfekci firmy Continental Barum s.r.o.
CHTT	Continental HT Tyres, s.r.o.
JIT	Just-in-time.
KM	Typ stroje na konfekci firmy Continental Barum s.r.o.
LIFO	Last In, First Out.
MRP	Material Requirements Planning.
PLT	Osobní a lehké nákladní automobily.
PROSI	PROduction Planning, Scheduling and Information.
PRT	Perturbace.
SOMA	SamoOrganizující se Migrační Algoritmus.
TOC	Theory of Constraints.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PI CD-ROM

PŘÍLOHA P I: CD-ROM

Na CD je uložena diplomová práce ve formátu PDF/A.

Dále jsou zde dvě aplikace ve formátu .xlsm. Pro každé řešení problému rozvrhování surových plášťů na jednom stroji byl vytvořen samostatný soubor.