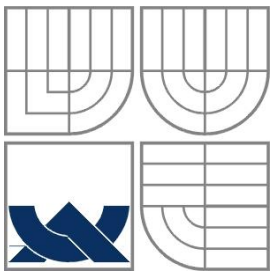
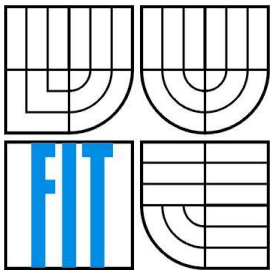


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTIMALIZACE ROZMÍSTĚNÍ DÍLŮ VE SKLADU

OPTIMISATION OF THE COMPONENTS LAY-OUT IN STOCK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR PYSZKO
AUTHOR

VEDOUCÍ PRÁCE — MGR. PETR ŠKODA
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá optimalizací rozmístění dílů ve skladu. Řešení tohoto problému je založeno na čtyřech metodách rozmístění, jejichž klady a zápory jsou diskutovány. Efektivita těchto metod je testována pomocí simulátoru vychystávání. V rámci práce byl navržen a následně implementován simulátor a optimalizátor skladu v jazyce C# za použití nástrojů, které poskytuje .NET framework.

Abstract

This thesis deals with optimisation of the components lay-out in stock. The solution of this problem is based on four methods of storage assignment whose pros and cons are discussed. The efficiency of these methods is tested by order picking simulator. Within this thesis, the stock simulator and optimizer has been designed and implemented using C# programming language and .NET framework tools.

Klíčová slova

optimalizace skladu, metody rozmístění zboží, skladování, simulátor vychystávání, C#, .NET

Keywords

warehouse optimalisation, storage assignment, warehousing, order picking simulator, C#, .NET

Citace

Pyszko Petr: Optimalizace rozmístění dílů ve skladu, bakalářská práce, Brno, FIT VUT v Brně, 2013

Optimalizace rozmístění dílů ve skladu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Petra Škody. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Petr Pyszko

5. 5. 2013

Poděkování

Rád bych poděkoval vedoucímu této práce Mgr. Petru Škodovi za ochotu a cenné rady. Dále bych chtěl poděkovat Radimu Fojkesovi, Ing. Antonovi Svrčkovi a Jiřímu Hermannovi za přínosné konzultace ve firmě Gates Hydraulics s.r.o.

Obsah

Kapitola 1
Úvod

Problém efektivního rozmístění zboží ve skladech řeší mnoho firem. V menších skladech se často používají obecné poučky o tom, že těžké, neobjemné a obrátkové zboží by mělo být blíže k vychystávacímu bodu. Vzhledem k velikosti dnešních obchodních a průmyslových skladů je však potřeba tento problém řešit pomocí výpočetních nástrojů.

Efektivita rozmístění zboží se projevuje zejména na rychlosti vychystávání zboží. V praxi se používají různé přístupy k rozmístění. Často, kvůli rychle se měnící poptávce, nemá optimalizace rozmístění smysl. Optimalizace nejčastěji vychází z historických dat, ze kterých získáváme znalosti o vychystávání v minulosti. Pokud je možné předpovídat poptávku, je vhodné tato data zahrnout do optimalizace.

Cílem této práce je návrh a implementace aplikace, která na základě historických dat optimalizuje rozmístění ve skladu firmy Gates Hydraulics. Pro ohodnocení kvality rozmístění bude implementován simulátor vychystávání. Optimalizace bude moci probíhat podle více algoritmů. Kvalitnější rozmístění by mělo zajistit snížení nákladů na provoz skladu.

Ve druhé kapitole je vysvětlen pojem sklad a jeho důležitost v logistice. Dále se zde nachází popis skladových procesů a uvádí se přístupy k rozmístění zboží a metody optimalizace. Kapitola číslo 3 popisuje sklad firmy Gates z hlediska dělení a pohybu materiálu. V následující kapitole se nachází analýza a návrh aplikace – kapitola je zaměřena na výpočet cesty mezi lokacemi, hodnocení dostupnosti lokací a přístupy k optimalizaci rozmístění. V druhé části čtvrté kapitoly se pak nachází návrh uživatelského rozhraní aplikace a její začlenění do systému firmy. Kapitola číslo 5 je věnována implementaci simulátoru a optimalizátoru. Na konci práce jsou shrnuty dosažené výsledky a poté následuje závěr, ve kterém jsou zhodnoceny výsledky práce a navrhnuta možná vylepšení.

Kapitola 2
Skladování

Skladování je nedílnou součástí každého logistického systému. Jedná se o článek mezi výrobcem a spotřebitelem. Tento článek vyrovnává výkyvy ve výrobě a poptávce po zboží. Sklad bývá využíván v různých částech logistického řetězce. Může se nacházet u výrobce, distributora či koncového zákazníka. Dle literatury tvoří skladování až 20 % celkových nákladů za logistiku. [2]

2.1 Sklady

2.1.1 Využití skladů

Náklady na skladování zahrnují využití prostorů, provoz skladové techniky, vybavení a v neposlední řadě také náklady za pracovní sílu. Přesto jsou sklady přínosným logistickým prvkem, a to zejména z těchto důvodů [9]:

- dosahování efektivního transportu (např. plné využití nákladního prostoru vozidla, sdružování zásilek do jednoho transportu);
- připravenost na poptávku (výroba by nestačila pokrýt poptávku, která je předvídatelná);
- využití zlevněného zboží a jeho zakoupení do zásoby;
- vyrovnání se s proměnlivými podmínkami na trhu (sezónnost zboží, výkyvy poptávky);
- překonání časových rozdílů mezi producenty a spotřebiteli;
- podpora „just-in-time“ přístupu k výrobě;
- sjednocení objednávek zákazníkovi do jedné;
- uložení dočasného materiálu k likvidaci či recyklaci.

2.1.2 Základní procesy ve skladu

Operace ve skladu lze rozdělit do těchto částí [2]:

- **Příjem,** který zahrnuje vyložení materiálu od dodavatele na příslušné vstupní místo. Následuje inspekce zboží a hledání zmetků.
- **Naskladňování,** což je proces, kdy je materiál vkládán na určené místo ve skladu a poté je aktualizován záznam o existenci dílu ve skladu.
- **Vychystávání,** které zahrnuje sběr sady dílů ze skladu pro uspokojení potřeb zákazníka nebo k připravení materiálu k dalšímu zpracování.
- **Expedice zboží**, což je děj, při kterém je materiál nakládán do přepravních vozidel a odesílán zákazníkovi. Zahrnuje také výslednou inspekci a balení zboží. Databáze skladu je aktualizována.

2.1.3 Dělení skladů

Sklady lze dělit dle mnoha kritérií. Dle konstrukce, dle průtoku zboží, dle vlastnictví, dle funkce, dle uskladnění zboží a dle míry automatizace [14]. Z hlediska této práce nejsou tato dělení příliš důležitá

kromě dělení dle míry automatizace, které bude představeno v následující podkapitole z pohledu vychystávacích systémů.

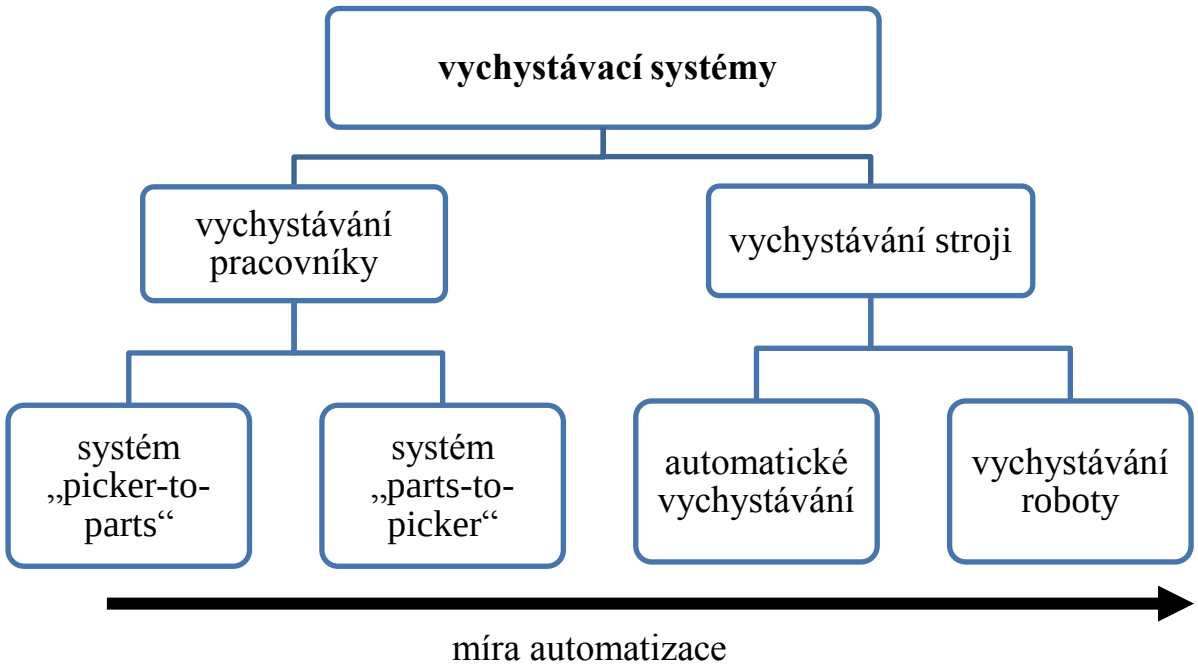
2.2 Vychystávání zboží

Vychystávání zboží je proces ve skladu, kdy je potřeba vyskladnit určité typy zboží v zadané kvantitě. Děje se tak po předchozím požadavku zákazníka. Oproti procesu naskladňování je objem přeneseného materiálu mnohem menší. Tento proces je kritický pro každý logistický řetězec, protože na jeho efektivitě, rychlosti a kvalitě závisí úroveň zákaznického servisu a také značná část nákladů na provoz skladu.

Ze všech procesů ve skladu je tento považován za nejvíce nákladný. Tvoří 50 až 60 % celkových nákladů za skladování v závislosti na efektivitě a míře automatizace skladových operací. Tento vysoký podíl na celkových nákladech je způsoben zejména faktem, že velká míra skladů není v tomto procesu automatizována a vyskladňování provádí pracovníci ručně. [5]

2.2.1 Klasifikace vychystávacích systémů

Systémy vychystávání lze dělit do mnoha kategorií dle míry automatizace. Základní dělení se odvíjí od toho, zda je proces vychystávání prováděn pracovníky nebo stroji. Plně automatizované sklady jsou zatím jen velice zřídka využívané. Systémy, kde vychystávají pracovníci, se dělí na „pickers-to-parts“ a „parts-to-picker“. Do češtiny by se tyto pojmy daly přeložit jako sběrači k dílům a díly ke sběračům. V práci budou využity anglické výrazy, jelikož výskyt českých ekvivalentů nebyl v literatuře zaznamenán.



Obrázek 2.1: Klasifikace vychystávacích systémů dle míry automatizace (vychází z [5])

Systémy **„picker-to-parts“** obsahují pracovníka, který prochází (popř. jede) skladem a vychystává jednotlivé objednávky. Pracovník většinou používá ruční vozík a na něj si skládá potřebné typy zboží, které jsou nutné ke kompletaci objednávky. Pracovník také může při jedné cestě sdružovat objednávky, čímž sníží celkovou cestu oproti vychystávání zvlášť. Tyto systémy se pak dále dělí dle úrovně

vychystávaného zboží. U nízko úrovňových (low-level) systémů je zboží přímo dosažitelné rukama skladníka. U vysoko úrovňových (high-level) systémů je pak třeba vysokozdvižného vozíku nebo jeřábu pro přístup k vysoko uloženému zboží.

V systémech **„parts-to-picker“** vyskladní automatické zařízení potřebné zboží na určité místo, kde pracovník odebere potřebné množství. Zbytek nevyužitého zboží je pak navráceno zpátky na svou určenou pozici. Tato technologie automatického uložení a vychystávání zboží je známa pod pojmem AR/RS (Automated Storage and Retrieval Systems). V těchto systémech je značně eliminován čas potřebný k vychystání, protože pracovníci nemusí procházet sklad a zboží hledat. Počáteční náklady na výstavbu tohoto skladového systému však mnohonásobně převyšují náklady na systém „picker-to-parts“.

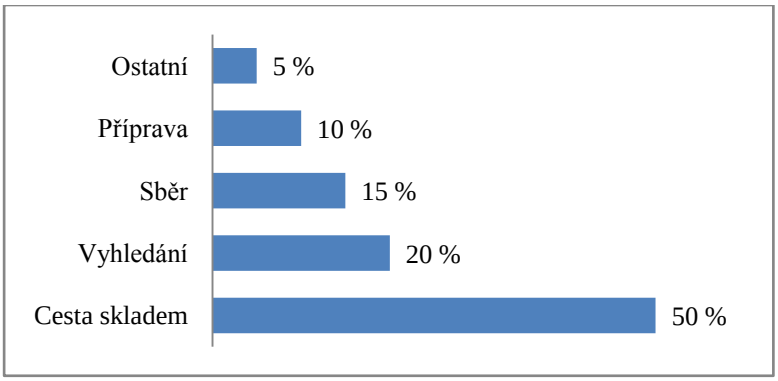
Automatické a robotické vychystávání je použito pouze ve výjimečných případech (např. cenné, velmi malé nebo velmi křehké zboží).

Tato práce se bude zabývat především systémy „picker-to-parts“. Dle literatury je ve více než 80 % skladů ve východní Evropě použit právě tento vychystávací systém. [2]

2.2.2 Proces vychystávání

Při vychystávání dostane pracovník seznam zboží v požadované kvantitě na vychystávacím listě. Pracovník použije ruční vozík a následně prochází skladem a vyjímá zboží v požadovaném množství. Poté se vrátí na místo, odkud přišel, a zboží vyloží. Tento děj lze rozdělit do následujících časových intervalů [5]:

- **čas cesty**, tj. čas potřebný k přesunu pracovníka ze startovního bodu k první lokaci s potřebným zbožím, mezi lokacemi se zbožím a zpět ke startovnímu bodu;
- **čas hledání**, tj. čas, který pracovník stráví hledáním lokace s požadovaným zbožím;
- **čas sběru**, tj. doba, za kterou pracovník naloží zboží do přepravního zařízení (např. ruční paletový vozík);
- **čas přípravy**, čas potřebný k administrativě týkající se převzetí vychystávacího listu a přípravě přepravního zařízení před cestou.



Obrázek 2.2: Obvyklé rozložení času při vychystávání [5]

2.3 Optimalizace procesu vychystávání

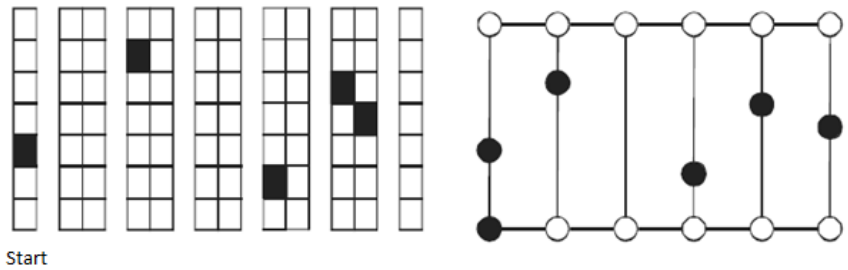
Jak již bylo uvedeno, vychystávání tvoří velkou část nákladů na skladování zboží. Proto je vhodné tuto činnost optimalizovat, a zvýšit tak efektivitu vychystávání. Rychlejší vychystávání zlepšuje úroveň zákaznického servisu a snižuje náklady na provoz skladu.

Optimalizace této činnosti je komplexní záležitost. Často vyžaduje vytvoření detailního modelu zahrnujícího dovoz zboží, pohyb skladníků, směrovací metodu, způsob ukládání zboží a zákaznickou poptávku.

2.3.1 Průchod skladem

Účelem optimalizace cesty skladem je seřazení jednotlivých položek vychystávacího listu tak, aby výsledná cesta byla co nejkratší. Problém směrování pracovníka skladem je speciálním případem *problému obchodního cestujícího*. Jedná se o obtížný optimalizační problém, který hledá nejkratší cestu mezi *n* městy tak, aby každé město bylo navštíveno právě jednou.

Obrázek 2.1 demonstruje podobnost tohoto problému. Na levé straně se nachází schéma skladu s vyznačenými lokacemi, které je nutno navštívit. Na pravé straně se pak nachází graf, který skladu z hlediska cest odpovídá.



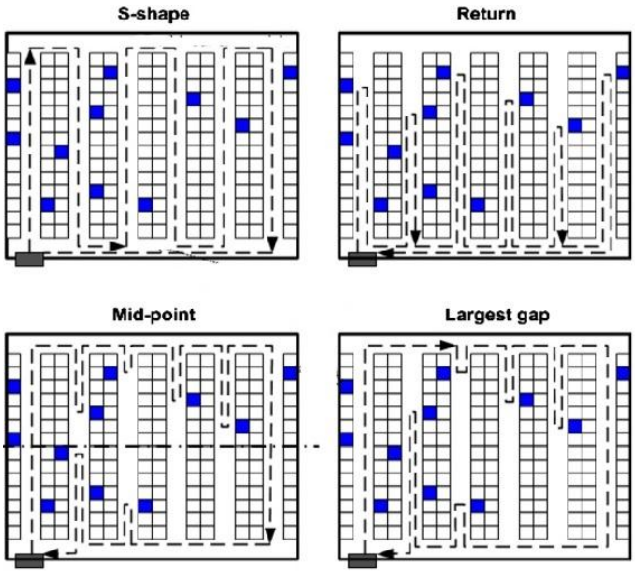
Obrázek 2.1: Graf cesty skladem [2]

Existují zde určité odlišnosti oproti klasickému problému obchodního cestujícího. V grafu se nacházejí uzly, které nemusí, ale mohou být navštíveny (bílé uzly). Tyto uzly se nacházejí na konci uliček mezi regály. Uzly, jež jsou označeny černou barvou a je nutné je navštívit pro kompletaci objednávky, mohou být navštíveny i vícekrát. Tento problém je klasifikován jako *Steinerův problém obchodního cestujícího* a není řešitelný v polynomiálním čase. [2]

2.3.2 Heuristické metody průchodu

V praxi se jen velice zřídka prochází sklad optimální cestou. I kdyby sklady disponovaly technologií, která by pracovníkovi nabídla optimální cestu, sledování navržené cesty by ho zdržovalo. Pracovníkovi by také cesta navržená systémem mohla připadat nelogická, a tak by ji ignoroval. Optimální cesta navíc nepočítá s možností zablokování uličky při paralelní práci více pracovníků.

Heuristické metody jsou jednodušší a pracovník si je rychle osvojí. Při pokusech s náhodnými sklady a náhodně generovanými objednávkami nedosahovaly výrazně horších výsledů oproti optimálním cestám [3]. Některé heuristické metody minimalizují riziko blokování uliček. V obrázku 2.3 jsou uvedeny nejčastější heuristické metody průchodu skladem.



Obrázek 2.3: Heuristické metody směřování při vychystávání [11]

Při metodě **tvaru s** (s-shape) se prochází skladem v jednom směru a ulička je procházena skrz, pouze když se v ní nachází alespoň jedna položka k vychystání. Tato metoda snižuje riziko blokování uliček, pokud je hustota sběrových lokací vysoká. Další metodou je **metoda návratu** (return), při které pracovník vstupuje do uličky i vystupuje z uličky stejnou cestou. Tato metoda je nezbytná v případě, že uličku nelze na druhém konci opustit z bezpečnostních důvodů, nebo v případě, že je ulička slepá. **Středová metoda** (midpoint) rozděluje sklad na dvě poloviny. Při průchodu zadní příčnou uličkou je vyskladněno pouze zboží, které se nachází v zadní polovině skladu, a při zpětném průchodu přední příčnou uličkou pak zboží v přední polovině. Strategie **největší mezery** (largest-gap) pak určuje, kdy se pracovník v uličce navrací na základě největší mezery. Největší mezera se vybere z mezer mezi začátkem uličky a první sběrovou lokací, mezi dvěma lokacemi za sebou v uličce, nebo poslední sběrovou lokací a koncem uličky. Pokud pracovník dorazí k největší mezeře, navrací se zpět na místo, odkud do uličky vstoupil. Pokud v uličce zbylo nějaké zboží určené k vyskladnění, pracovník jej sebere po zpáteční cestě. Tato metoda efektivně minimalizuje cestu potřebnou k vyskladnění, je však oproti ostatním poměrně složitá. [11]

Tyto heuristické metody jsou vhodné pro sklady, ve kterých se skladník pohybuje pěšky nebo používá při vyskladňování výhradně vozidlo, uličky mezi regály jsou přístupné z obou stran a zboží je umístěno vždy na jedné lokaci. V případě, že by se zboží nacházelo ve více lokacích, musela by být heuristická metoda složitější a vybírat nejvhodnější lokaci. Také při kombinování použití vozidla a chození pěšky pro zboží by heuristika měla zvažovat poměry rychlosti vozidla a rychlosti chůze. I náklady na provoz vozidla by měly být brány v úvahu při stanovování optimální cesty.

2.3.3 Sdružování objednávek

V některých skladových systémech se při vychystávání sdružují objednávky tak, aby se dosáhlo vyšší efektivity. Problém správného a efektivního sdružování je znám jako **„order batching problem“**. Existují algoritmy, které tento problém řeší a jsou popsány v [5]. Při zavedení sdružování objednávek v praxi je pak nutno přidat další proces třídění před expedicí objednávek zákazníkům. Je však dokázáno, že efektivní sdružování výrazným způsobem snižuje náklady na proces vychystávání. V této práci se však tímto tématem dále zabývat nebudeme.

2.4 Strategie rozmístění zboží ve skladu

Rozmístění zboží hraje při optimalizaci významnou roli. Správné rozmístění vede k rychlejšímu vyskladňování. Rozhodnutí, kam zboží umístit, by mělo být učiněno při jeho naskladňování.

Za účelem zrychlení procesu vychystávání je v mnoha případech vhodné rozdělit sklad do dvou částí. **Zásobní část** (reserve area) je ve skladu dál od místa, kam se vyskladňuje. Tato část je určena pro velké množství zboží, které se naskladňuje. **Sběrová část** (forward area) je umístěna blíže k vyskladňovacímu bodu. Z těchto míst se zboží vyskladňuje pro kompletaci objednávek. Pokud dojde k vyčerpání zboží z této části, doplní se ze zásobní části.

2.4.1 Metody rozmístění

Existuje mnoho způsobů, jak zboží ve skladu rozmísťovat. Mezi nejpoužívanější v praxi patří tyto metody [2]:

Náhodné uložení (random storage)
Při náhodném uložení je každá příchozí paleta (nebo jakékoliv množství téhož produktu) náhodně umístěna do některé z aktuálně volných pozic ve skladu. Tato metoda je efektivní z hlediska vysokého využití prostoru skladu, avšak zvyšuje délky cest při vychystávání. Metoda je použitelná pouze ve skladových systémech, které jsou pod kontrolou počítačovým systémem (hledání zboží v náhodném skladu by enormně zpomalovalo proces).

Ukládání na nejbližší volnou pozici (closest open location storage)
Pokud určují pozici nově příchozího zboží pracovníci, je velice pravděpodobné, že se bude jednat právě o toto rozmístění. Tato strategie většinou vede k přeplnění skladu u místa s příjmem zboží a prázdnému skladu na konci. Efektivita je srovnatelná s náhodným uložením za předpokladu, že manipulace probíhá pouze s plnými paletami zboží.

Vyhrazené rozmístění (dedicated storage)
Každé zboží má vyhrazený prostor ve skladu, i když zrovna není k dispozici. Rozmístění je výhodné, protože si skladníci zapamatují pozice zboží, a proto je vyhledávání rychlejší. Toto rozložení umožňuje také umisťovat často dohromady vychystávané zboží k sobě. Nevýhodou je nízké využití prostoru skladu. Rozmístění je vhodné aplikovat na sběrovou část skladu (forward area) a v zásobní ponechat např. náhodné uložení.

Rozmístění na základě obratu (full-turnover storage)
Zboží s vysokým obratem je uloženo blíže k vyskladňovacímu bodu. Zboží se při této metodě často hodnotí pomocí **COI** indexu (Cube per Order Index) [7]. Jedná se o poměr potřebného místa k udržování potřebného množství zboží a počtu objednávek, které toto zboží obsahují. Zboží s nižším indexem je umisťováno blíže. Toto rozmístění výrazně snižuje délky cest při vychystávání. Nevýhoda spočívá v tom, že je potřeba detailně znát kvantitu příchozího a odchozího zboží a mít přesné statistiky o pohybech zboží ve skladu. Strategie ztrácí efektivnost, pokud dochází ke změně poptávky, a sklad je pak nutno obtížně přeorganizovat tak, aby odpovídal novým požadavkům.

Rozmístění dle tříd (class-based storage)
Tento přístup kombinuje metody uvedené výše. Zboží je rozděleno do tříd na základě nějakého měřitelného statistického ukazatele (např. COI, popularita atd.). Jednotlivé třídy mají ve skladu vyhrazené

místo a zboží je v nich umísťováno náhodně. Tato metoda není tak efektivní jako rozmístění na základě obratu, avšak její realizace je podstatně jednodušší.

Rozmístění dle shluků (family grouping)
Výše uvedené metody nebraly v potaz fakt, že se v objednávkách mohou vyskytovat skupiny zboží, které jsou vychystávány často společně. Tyto typy zboží, nacházející se v častých shlucích, je vhodné umisťovat ve skladu blízko u sebe. Metodu je vhodné kombinovat s některou z již zmíněných strategií. Například je možné shluky umisťovat do tříd na základě kombinace jejich statistických vlastností.

Tento přístup k rozmístění je nejsložitější a časově nejnáročnější. Zahrnuje vyhledávání shluků (např. metodou apriori [15]), hodnocení shluků a jejich vhodné rozmísťování. Tento problém úzce souvisí s použitou metodou směřování ve skladu. Vzhledem k velikosti skladů použitých v praxi není možné nalezení optimálního řešení v reálném čase.

2.4.2 Matematická formulace kvality rozmístění

Pro vytvoření simulačního programu, který bude hledat optimální rozmístění skladu, je potřeba rychle vyčíslit kvalitu rozmístění. Ta se dá posuzovat podle dvou hodnot – blízkost často vychystávaných položek od místa vyskladňování a blízkost dvojic často dohromady vychystávaných položek. [8]

Skóre dle popularity zboží

$$\sum_{i=1}^{n} \frac{objednávky(p_i)}{|lokace(p_i)|} \cdot \sum_{s \in lokace\ (p_i)} vzdálenost(s, počátek)$$

Rovnice 2.1: Skóre na základě popularity zboží [8]

Skóre dle blízkosti častých dvojic zboží

$$\sum_{i=0}^{n} \sum_{j=i+1}^{n} \frac{častost(p_i, p_j)}{|lokace(p_i)| \cdot |lokace(p_j)|} \cdot \sum_{s_k \in lokace\ (p_i)} \sum_{s_l \in lokace\ (p_j)} vzdálenost(s_k, s_l)$$

Rovnice 2.2: Skóre dle vzdálenosti častých dvojic [8]

n	počet všech typů zboží
objednávky(p_i)	počet objednávek, ve kterých se vyskytuje produkt p_i
lokace(p_i)	množina lokací, ve kterých se nachází produkt p_i
vzdálenost(s_i,s_j)	vzdálenost lokací s_i a s_j ve skladu
častost(p_i,p_j)	počet objednávek, ve kterých se p_i a p_j vyskytují společně

Skóre dle popularity zboží je číslo, které určuje kvalitu rozmístění na základě vzdálenosti často vychystávaných dílů od místa, kam se vyskladňuje. Při takovém rozmístění zboží, kdy nejčastěji vychystávané díly jsou nejpřístupnější, je toto skóre nejnižší (rozmístění dle popularity). **Skóre dle blízkosti dvojic** určuje kvalitu rozmístění dle vzájemné blízkosti často dohromady vychystávaných dvojic zboží. Pokud jsou často dohromady vychystávané dvojice umístěny ve skladu blízko sebe, toto skóre je nejnižší.

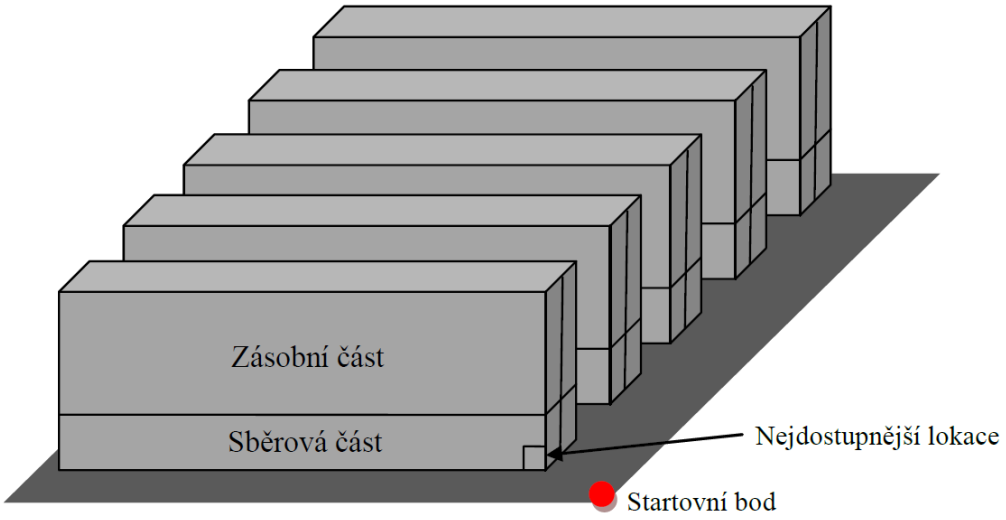
Celkové skóre je pak váženým součtem těchto dvou hodnot. Vyvážení těchto dvou skóre je potřeba experimentálně určit pro konkrétní sklad. Toto rychlé vyčíslení kvality umožňuje hledání optimálního skladu pomocí heuristických metod. Mezi nejčastěji používané patří metoda lokálního hledání a metoda simulovaného žíhaní. [8]

Kapitola 3
Sklad ve firmě Gates Hydraulics s.r.o.

Sklad firmy Gates Hydraulics je určen pro dočasné uložení dílů určených pro následné zpracování. Sklad je tvořen pěti regálovými konstrukcemi, které jsou přístupné z obou stran. Palety s materiálem jsou uloženy celkem v osmi výškových úrovních. Do jedné regálové konstrukce lze na délku umístit 45 plných palet. Celkově se tedy ve skladu nachází 3600 paletových lokací.

3.1 Rozdělení skladu

Lokace nacházející se výš než v druhé úrovni nejsou ručně dostupné a je nutné použití vysokozdvižného vozíku. Ve firmě Gates Hydraulics je vždy výhodnější vyskladnit díl ručně než použít vysokozdvižný vozík, a to i v případě, že by se díl nacházel v nejnedostupnější lokaci. Vysokozdvižný vozík použitý ve firmě Gates Hydraulics se pohybuje velice rychle v rámci uličky, avšak přejíždění mezi nimi je z bezpečnostních důvodů velice pomalé. Výše položené lokace tvoří zásobní část skladu, kde se nachází větší množství stejného dílu. Níže položené lokace jsou pak využity jako sběrová část z důvodu lepší dostupnosti.



Obrázek 3.1: Rozdělení skladu

3.2 Manipulace s díly

Stav sběrové části skladu před optimalizací víceméně odpovídá náhodnému rozmístění. Při snížení stavu zásob jednoho dílu dojde k automatickému objednání většího množství a to je naskladněno do zásobní části. Při naskladňování jsou díly nejčastěji uloženy nad lokaci, kde je stejný typ dílu ve sběrové části. Pokud se lokace ve sběrové části uvolní, vysokozdvižný vozík lokaci naplní dílem uloženým výše v zásobní části.

Kapitola 4
Analýza a návrh aplikace

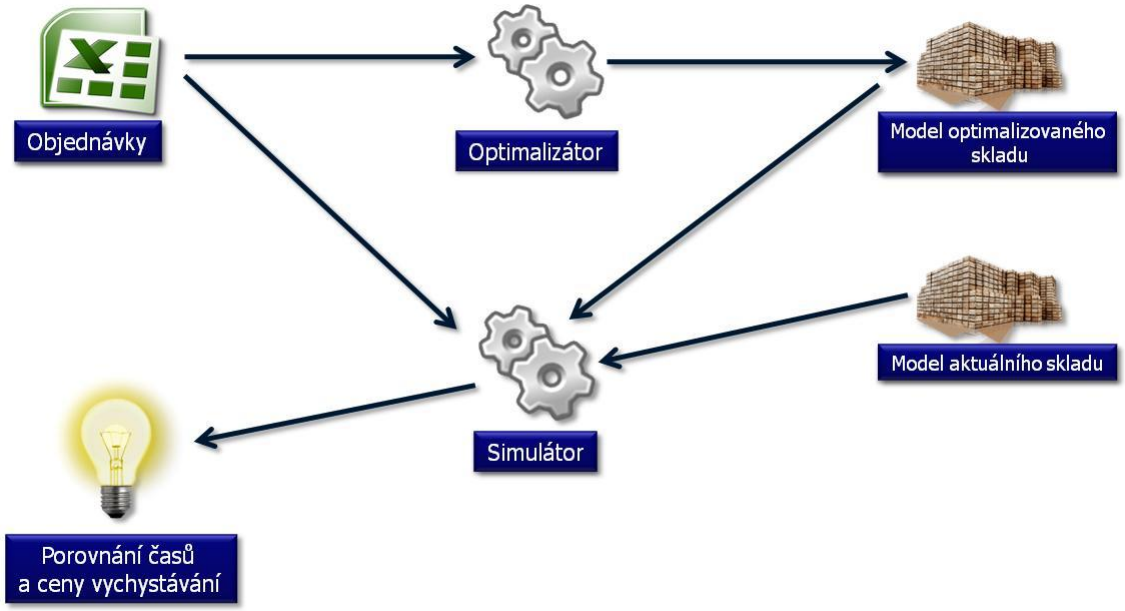
Optimalizace rozmístění dílů ve skladu je náročný proces, který bude zahrnovat implementaci simulátoru vychystávání a také optimalizačního algoritmu. Jak již bylo zmíněno v kapitole 2.4.1, v praxi nelze najít optimální řešení v reálném čase.

Tato kapitola se bude zabývat možnými přístupy k hledání lépe optimalizovaného skladu pro účely skladu firmy Gates Hydraulics s.r.o. V druhé části se pak zaměříme na analýzu požadavků, které by měla výsledná aplikace splňovat a její začlenění do systému firmy.

4.1 Zadání

Cílem této práce je vytvoření desktopové aplikace v jazyce C#, která bude spolupracovat se skladovým systémem firmy. Aplikace bude umožňovat optimalizaci modelu aktuálního skladu a porovnat ho z hlediska efektivity vychystávání se skladem aktuálním. Na základě výstupu tohoto porovnání by měl být uživatel schopen rozhodnout, zda se vyplatí sklad optimalizovat, optimalizace totiž zahrnuje složitou reorganizaci skladu.

Obrázek 4.1 znázorňuje schéma výsledného systému. Cílem optimalizátoru bude navrhnout model optimalizovaného skladu na základě historických dat. Simulátor pak dle historických objednávek vyčíslí efektivitu modelu skladu. Postup optimalizace a následné simulace je použit například v [8] a [1].



Obrázek 4.1: Schéma výsledného systému

4.2 Simulace vychystávání

Simulace vychystávání je proces, kdy simulátor na základě historických objednávek stanoví výslednou délku cesty, kterou pracovníci musí absolvovat pro vyskladnění zadaných dílů z modelu skladu. Ve skladu firmy Gates je počet druhů zboží, jež je vychystáváno během jedné cesty, poměrně nízký (průměrně 2 až 3 položky), proto je zde možné použít algoritmus hrubé síly a nalézt optimální cestu.

Při vychystávání pracovníci nechodí vždy úplně optimální cestou, avšak vzhledem k nízkému počtu zboží najednou se absolvovaná cesta blíží cestě optimální. Pro účely porovnávání optimalizovaného modelu skladu bude program počítat s optimálními cestami skladníků.

Vstupem simulátoru vychystávání bude libovolný model skladu a historické záznamy o vychystávání. Program simuluje proces vychystávání a výstupem bude statistika zahrnující délku všech cest skladníka a využití vysokozdvižného vozíku. Na základě této podrobné statistiky uživatel zhodnotí kvalitu rozmístění zboží ve skladu.

4.2.1 Hledání nejkratší cesty

K nalezení nejkratší cesty při vychystávání nízkého počtu dílů není potřeba heuristických metod pro řešení problému obchodního cestujícího. Mezi tyto metody patří například *2-aproximační* či *Christofidesův* algoritmus, které tento problém řeší v polynomiálním čase. I v nejhorším případě šesti dílů v objednávce ve firmě Gates Hydraulics jsou dnešní počítače schopné otestovat všechny permutace lokací ve skladu a nejkratší cestu nalézt v relativně krátkém čase.

Pro hledání nejkratší cesty je nutné mít funkci, která stanoví vzdálenost mezi libovolnými dvěma lokacemi ve skladu. Na základě znalosti parametrů skladu není problém funkci navrhnout a později implementovat. V obrázku 4.2 je znázorněna cesta mezi dvěma lokacemi X a Y. Jak je vidět, výpočet nebere v úvahu šikmé pohyby pracovníka, což ve výsledku není příliš podstatné. Cestu lze rozdělit do dvou částí – na horizontální a vertikální. Jejich součet pak tvoří délku výsledné cesty. Výpočet celkové cesty je popsán rovnicemi počínaje rovnicí 4.1. Pro vyšší přesnost lze hodnotu korigovat dle pozice zboží v daném regálu, čímž se však zabývat nebudeme. Tento výpočet je platný pouze pro případy, kdy pracovník přechází mezi uličkami. Délku cesty v rámci jedné uličky není problém stanovit jednoduchým výpočtem.

Obrázek 4.2: Cesta mezi dvěma lokacemi

Horizontální cesta

Horizontální cestou se rozumí vzdálenost, kterou pracovník vykoná v rámci uliček mezi řadami (regálovými konstrukcemi). Tuto vzdálenost lze vypočítat dle níže uvedené rovnice. Pokud je druhá lokace v jiné řadě, pracovník může jít levou i pravou stranou. Simulátor započítá vždy kratší variantu. Tento výpočet vychází z číslování regálů zleva doprava.

Cesta levou stranou:

$$horizontální\ délka = (regál(X) + regál(Y)) \cdot šířka\ regálu$$

Rovnice 4.1: Výpočet vertikální cesty levou stranou

Cesta pravou stranou:

$$horiz.délka = \Big(\big(počet\ regálů - regál(X)\big) + \big(počet\ regálů - regál(Y)\big) + 2\Big) \cdot šířka\ regálu$$

Rovnice 4.2: Výpočet vertikální cesty pravou stranou

Vertikální cesta

Pokud pracovník přechází mezi uličkami, mluvíme o vertikální cestě. Tuto vzdálenost nelze jednoduše vyčíslit z rozdílu řad, mezi kterými prochází, protože mezi některými se nachází ulička. Výpočet je kvůli lepší přehlednosti rozdělen do dvou rovnic v závislosti na tom, zda je řada s nižším pořadovým číslem sudá či lichá. To lze zjistit pomocí rovnice 4.3.

$$\Delta řad = |řada(X) - řada(Y)|$$

Rovnice 4.3: Výpočet rozdílu řad

Lichá nižší řada:

$$minimum\big(řada(X), řada(Y)\big)\ mod\ 2 = \mathbf{1}$$

Rovnice 4.4: Rovnice pro zjištění, zda je nižší řada lichá

$$vertikální\ délka = \left\lfloor \frac{\Delta řad}{2} \right\rfloor \cdot šířka\ uličky + \ (\Delta řad - \left\lfloor \frac{\Delta řad}{2} \right\rfloor) \cdot šířka\ dvou\ řad$$

Rovnice 4.5: Výpočet délky vertikální cesty pro lichou nižší řadu

Sudá nižší řada:

$$minimum\big(řada(X), řada(Y)\big)\ mod\ 2 = \mathbf{0}$$

Rovnice 4.6: Rovnice pro zjištění, zda je nižší řada sudá

$$vertikální\ délka = \left\lfloor \frac{\Delta řad}{2} \right\rfloor \cdot šířka\ dvou\ řad + \ (\Delta řad - \left\lfloor \frac{\Delta řad}{2} \right\rfloor) \cdot šířka\ uličky$$

Rovnice 4.7: Výpočet délky vertikální cesty pro sudou nižší řadu

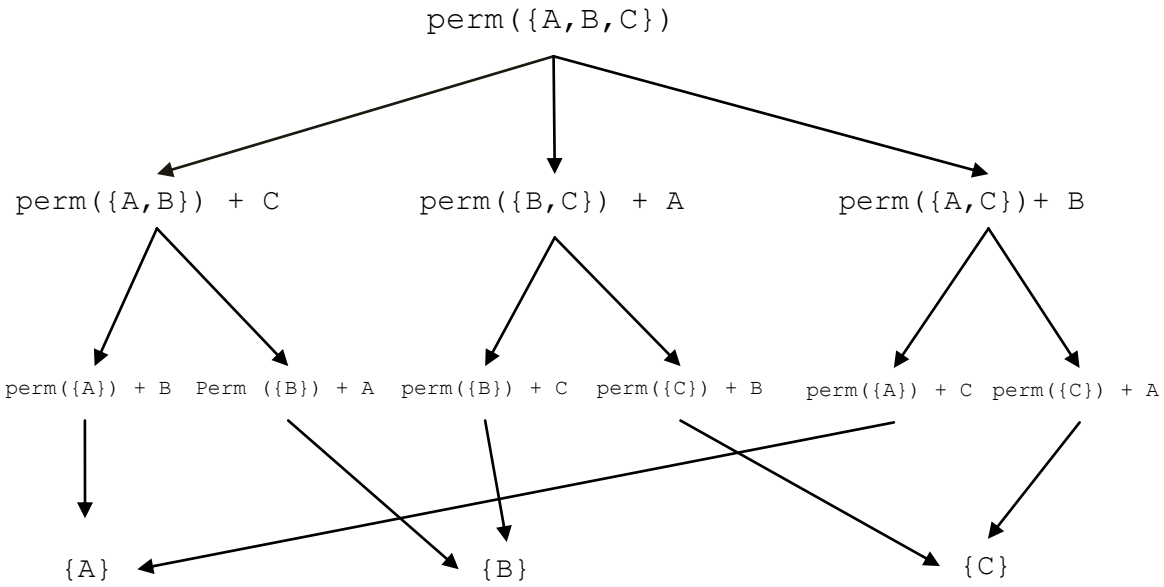
X, Y	startovní a konečná lokace
regál(X)	číslo regálu, ve kterém se nachází lokace X
řada(X)	číslo řady, ve které se nachází lokace X

4.2.2 Generování permutací lokací

Když už dokážeme vyčíslit délku cesty mezi dvěma lokacemi, můžeme hledat optimální cestu při vyskladňování dílů pro kompletaci objednávek. Nejdříve je ale potřeba vygenerovat všechny permutace lokací. Počet variant cest je roven faktoriálu počtu lokací, které je nutné obejít.

Příklad variant cest mezi lokacemi A, B, C:
{start, A, B, C, cíl} 66 m
{start, B, A, C, cíl} 45 m
{start, A, C, B, cíl} 40 m
{start, B, C, A, cíl} 40 m
{start, C, A, B, cíl} 45 m
{start, C, B, A, cíl} 66 m

Mezi variantami cest se nacházejí vždy dvojice se stejnou délkou cesty lišící se pouze směrem průchodu. Pokud je počet lokací k navštívení menší než tři, není nutné permutace generovat a libovolná cesta je i cestou optimální. Obrázek 4.3 znázorňuje princip rekurzivního generování všech permutací. Pro generování permutací existují i efektivnější metody, které jsou popsány v [12]. Pro účely simulátoru v této práci dostačuje i tato naivní metoda.



Obrázek 4.3: Rekurzivní generování permutací

4.3 Rozmístění dílů

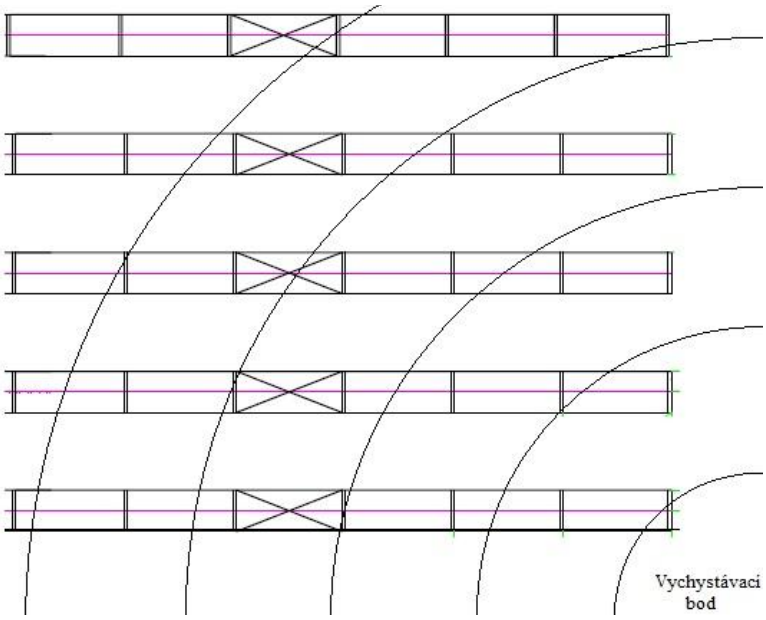
K vytvoření optimalizovaného modelu skladu je potřeba znát všechny dostupné lokace a všechny díly nutné ke kompletaci zadaných objednávek. Při optimalizaci můžeme vycházet z modelu aktuálního skladu a zvýšit jeho efektivitu nebo vytvářet úplně nový model. Optimalizace aktuálního skladu má značnou výhodu v tom, že když provedeme malé množství změn, optimalizace reálného skladu není tak nákladná. V případě aplikace nového modelu skladu je nutná plná reorganizace reálného skladu.

4.3.1 Hodnocení lokací skladu dle dostupnosti

Před samotným umisťováním dílů do lokací v modelu skladu je nutné znát výhodnost lokace. Obecně platí, že lokace, která je blíže startovnímu bodu, je výhodnější. To ovšem neplatí pro lokace, které nejsou přístupné ručně a kde je nutné použít vysokozdvižný vozík, jenž v případě firmy Gates Hydraulics nemá stejný startovní bod jako pracovník. Proto se výpočet skóre pro jednotlivé lokace rozpadne na dva případy – skóre pro ručně dostupné lokace (sběrová část skladu) a skóre pro vysoko položené lokace (zásobní část skladu). Nižší skóre znamená dostupnější lokaci.

Ručně dostupné lokace

Skóre lokací, které jsou ručně dostupné, odpovídá jejich vzdálenostem od startovního bodu pracovníka (viz obrázek 4.4). Tuto vzdálenost lze rozdělit na dvě části – cesta k nejdostupnější lokaci a cesta od ní k lokaci, pro kterou chceme skóre vypočítat. Cesta k nejdostupnější lokaci je předem známá z plánku skladu a pro výpočet cesty mezi dvěma lokacemi použijeme výpočet z kapitoly 4.2.1. Součet těchto hodnot je roven skóre pro danou lokaci.



Obrázek 4.4: Rozdělení lokací skladu do zón dle dostupnosti

Lokace dostupné vysokozdvižným vozíkem

Výpočet skóre uvedený níže ohodnocuje lokace na základě čísla sloupce, čísla řady (regálu) a výškové úrovně daného dílu se vzestupnou důležitostí. Výšková úroveň lokace má nejvyšší váhu ve výpočtu kvůli tomu, aby při zařazování dílů do lokací nebyly nejprve obsazeny lokace nad sebou, což by vedlo k nedostatku místa pro zásobní část v některých místech skladu.

Při výpočtu je výška a číslo řady násobeno koeficienty tak vysokými, aby při umisťování dílů podle vzestupného skóre byly nejprve obsazeny lokace, které se nacházejí v první řadě na nejnižší výšce (úrovni). K výsledné hodnotě skóre se následně přičte konstanta K, která představuje skóre pro nejnedostupnější lokaci, kterou lze vyskladnit ručně. Nyní je možné pro jakoukoliv lokaci ve skladu stanovit skóre, tudíž stanovit i její dostupnost.

$$skóré(L) = výška(L) \cdot 10000 + řada(L) \cdot 100 + regál(L) + K$$

Rovnice 4.8: Výpočet skóre pro ručně nedostupnou lokaci L

4.3.2 Rozmístění dílů na základě popularity

Tento přístup k rozmístění nevychází z modelu aktuálního skladu. Jedná se o aplikaci *COI indexu* se zanedbáním parametru potřebného prostoru ke skladování dílů. Tuto hodnotu nejsme schopni z historických dat zjistit. Vzhledem k tomu, že díly skladované ve firmě Gates Hydraulics, se velikostí nijak diametrálně neliší, lze tuto hodnotu zanedbat bez výrazného snížení kvality rozmístění.

Popularitou dílu rozumíme počet objednávek, které tento díl obsahují. Z databáze historických objednávek lze jednoduše SQL dotazem získat tabulku, ze které lze pro každý díl tuto hodnotu stanovit. Při rozmísťovaní se nejpopulárnější díly umísťují na nejlépe dostupné lokace ve skladu. Takovým způsobem postupujeme, až umístíme nejméně populární díl. Neobsazené lokace (méně dostupné) pak tvoří zásobní část skladu.

Tento způsob znatelně zlepší rozmístění oproti aktuálnímu skladu (téměř náhodné rozmístění). Mezi jeho výhody patří jednoduchost a rychlost. Rozmístění však nebere vůbec v potaz časté shluky zboží.

4.3.3 Rozmístění dle častých shluků

U tohoto přístupu k rozmístění je potřeba zjistit, zda existují v objednávkách určité vzory (shluky) dílů, které jsou vychystávány často společně. Tyto shluky musí splňovat minimální podporu (počet objednávek, ve kterých se shluk vyskytuje, je vyšší než minimální podpora).

Časté shluky umisťujeme blízko sebe postupně od těch nejčastějších. Tento přístup výrazně snižuje cestu skladem při vychystávání shlukových dílů. Problémem rozmístění je však to, že mnohé díly patří do více častých shluků. Také stanovení hodnoty, kdy je shluk považován za častý, není jednoduché a je potřeba tuto hodnotu experimentálně hledat.

Vzhledem k povaze objednávek ve firmě Gates není nutné hledat shluky, které mají více než dva díly. Časté shluky s více díly se v objednávkách neobjevovaly. Algoritmus je časově náročnější než rozmístění na základě popularity. Oproti heuristickým metodám je však stále podstatně rychlejší.

4.3.4 Lokální hledání

Tento přístup patří k heuristickým metodám hledání optimálního skladu. Jako heuristickou funkci použijeme výpočet skóre z rovnic v podkapitole 2.4.2. Výpočet funkce musí být dostatečně rychlý, aby bylo možné provést potřebné množství iterací k nalezení lépe optimalizovaného modelu skladu.

Výchozím stavem modelu může být aktuální sklad, ale vhodnější je použít např. rozmístění na základě popularity, čímž výchozí model připravíme do mnohem lepšího stavu a urychlíme tím optimalizaci. Optimalizační krok bude zahrnovat náhodnou výměnu pozic dvou dílů a následný výpočet heuristické funkce. Pokud se hodnota funkce oproti předchozímu stavu zlepší (nižší skóre odpovídá lepšímu rozmístění), přejde model do nového stavu. V opačném případě se model uvede do stavu před výměnou pozic a náhodně se generuje další výměna.

Při optimalizačním procesu dochází ke generování velkého počtu změn, avšak jen opravdu málo změn je přijato. Zvýšení výkonnosti by se mohlo dosáhnout, pokud by se vyměňovaly pouze díly, které se nacházejí v podobně dostupných lokacích (vhodné jen pokud je počátečním stavem rozmístění dle popularity dílů). Možností je také těmto výměnám přiřadit vyšší pravděpodobnost výskytu při generování. Těmito postupy bychom eliminovali neúčinné výměny (např. málo obrátkový díl v méně dostupných pozicích přesunutý na dobře dostupnou lokaci). Další vylepšení by mohlo přinést to, že bychom výměnám často obrátkových dílů přiřadili větší pravděpodobnost. Pro provádění větších změn modelu v jednom kroku je vhodné náhodně cyklicky vyměňovat N pozic dílů.

Tento přístup zoptimalizuje sklad mnohem lépe než předchozí metody. Jeho nevýhodou je velká časová náročnost, která je daná zejména výpočtem heuristické funkce po každém provedeném kroku.

4.3.5 Metoda zadaného počtu změn

Výše uvedené strategie rozmisťují díly ve skladu tak, že při aplikaci optimalizace do reálného skladu bude nutné změnit pozici téměř všech dílů. Tento přístup není příliš výhodný, pokud se dopředu ví, že se povaha objednávek může brzy změnit a rozmístění skladu se tím stane neaktuální.

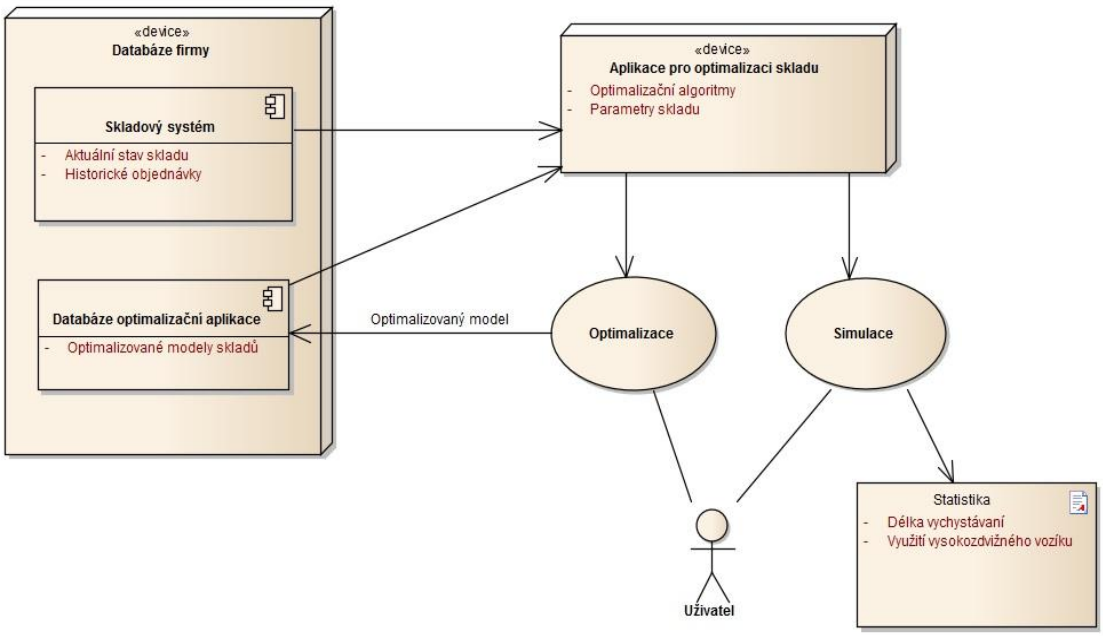
Metoda zadaného počtu změn vyhází z modelu aktuálního stavu a je možné nastavit, kolik změn se ve skladu má provést. Technicky metoda vychází z lokálního hledání. Při výhodném kroku se však změna neaplikuje, ale pouze zaznamená pro pozdější využití. V databázi změn se udržuje jen N nejvýhodnějších změn, které jsou aplikovány až po provedení zadaného počtu iterací. Výhodností změny rozumíme velikost snížení heuristické funkce po provedení iteračního kroku.

Tato metoda umožňuje znatelně vylepšit aktuální rozmístění, avšak oproti lokálnímu hledání není tak účinná. Hlavní výhodou je poměrně nenáročná aplikace změn do reálného skladu. Časová náročnost je stejná jako v případě lokálního hledání.

4.4 Návrh aplikace

Program pro optimalizaci skladu bude klasická desktopová aplikace. Přestože většina programů ve firmě jsou ASP.NET webové aplikace, je použití desktopové aplikace vhodné, a to zejména proto, že zatěžování firemního serveru časově náročnou optimalizací a simulací by přinášelo mnohé problémy. Rychlosti odezvy by byly nižší a muselo by se po síti přenášet značné množství dat. V neposlední řadě by byla implementace takové webové aplikace náročnější.

Historická data i model aktuálního skladu bude aplikace čerpat ze skladového systému firmy. Záznamy jsou poskytnuty pomocí databázových pohledů. Aplikace bude mít vlastní databázi, která je určena pro ukládání optimalizovaných modelů, z nichž se bude při reorganizaci skladu vycházet. Tyto uložené modely bude možno použít také pro simulaci na historických objednávkách. Aplikace umožní import historických objednávek i ze sešitů aplikace Microsoft Excel. Uložení optimalizovaného modelu bude možné do databáze, formátu XML a CSV[1]. Modely těchto formátů bude možné použít i při simulaci.



Obrázek 4.5: Schéma propojení aplikace s databázovým systémem firmy

[1] Formáty XML a CSV se používají pro textovou serializaci dat.

4.4.1 Návrh simulátoru

Simulátor bude komponenta, která na základě historických objednávek a modelu skladu určí celkové náklady za vychystávání. Bude se zde sledovat celková délka chůze pracovníků, počet vychystaných dílů, počet objednávek, průměrný počet dílů v objednávce a průměrná cesta skladem potřebná k vychystání jedné objednávky. Simulátor bude také zaznamenávat pohyb vysokozdvižného vozíku, který je nutno použít, pokud jsou díly uloženy vysoko v regálech. Zde jsou důležité délky cest, které vozík urazí v rámci uličky, počty přejíždění mezi uličkami, vertikální pohyb vidlice a celkový počet vychystaných dílů.

Na základě těchto statistických údajů bude možné stanovit náklady na provoz modelu skladu v zadaném časovém období. Nejdražší operací je výjezd vysokozdvižného vozíku při vyskladňování, proto je výhodné, aby se nejdůležitější díly nacházely ve sběrové části skladu.

Simulace je poměrně časově náročnou operací především v případech, pokud bude zadané časové období dlouhé a historické objednávky budou obsahovat desetitisíce objednávek. Výsledky simulace tedy nebude možné použít pro určování kvality rozmístění během optimalizace pomocí heuristických metod.

Simulace bude nejčastěji probíhat v časovém rozpětí tří měsíců nebo jednoho roku. Sledování průběhu delšího období by nebylo příliš přínosné, protože se charakter objednávek mění i vícekrát do roka. Rozmístění, které by vycházelo ze starších dat, by na budoucích objednávkách nemuselo být efektivní.

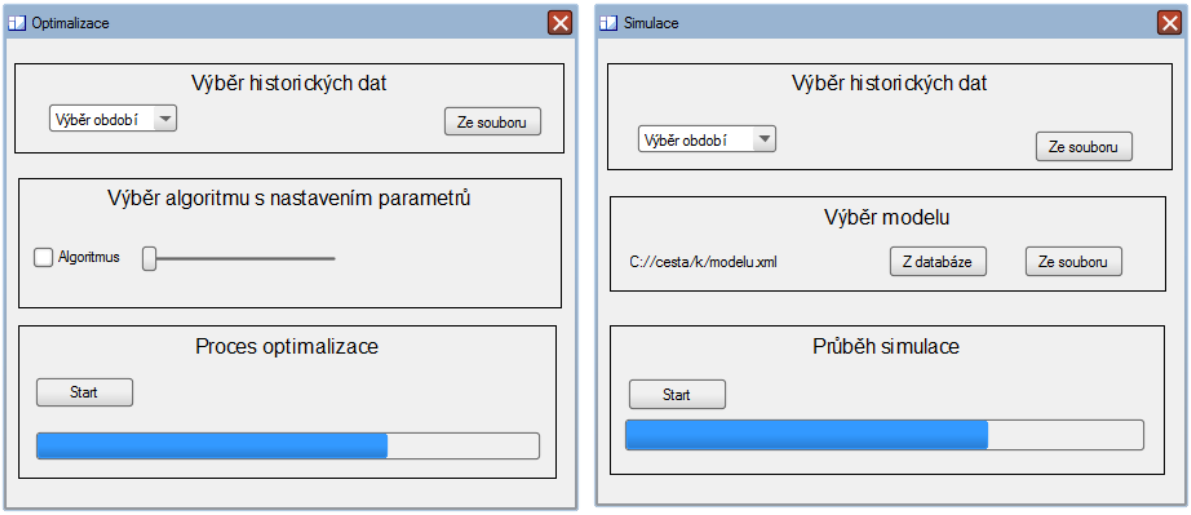
4.4.2 Optimalizátor

Tato část aplikace bude modifikovat výchozí model skladu dle jednoho z výše uvedených algoritmů. Výchozím modelem může být sklad aktuální nebo může být model vytvořen pouze ze znalosti charakteru historických dat. Optimalizace může probíhat dle více možných algoritmů, proto je vhodné vytvořit optimalizátor tak, aby se s jednotlivými algoritmy pracovalo stejným způsobem bez ohledu na to, který je využíván. Tohoto chování lze v jazyce C# dosáhnout buď pomocí dědičnosti, nebo pomocí rozhraní.

4.5 Grafické uživatelské rozhraní

Uživatelské rozhraní této aplikace by mělo být jednoduché a účelné. Uživatelé, kteří budou tuto aplikaci používat, bez problému zvládají základní práci s počítačem, takže by pro ně používání aplikace nemělo znamenat větší problém. Tito lidé jsou sice zvyklí požívat spíše webové aplikace, avšak, jak již bylo zmíněno, aplikace bude výhodnější desktopová.

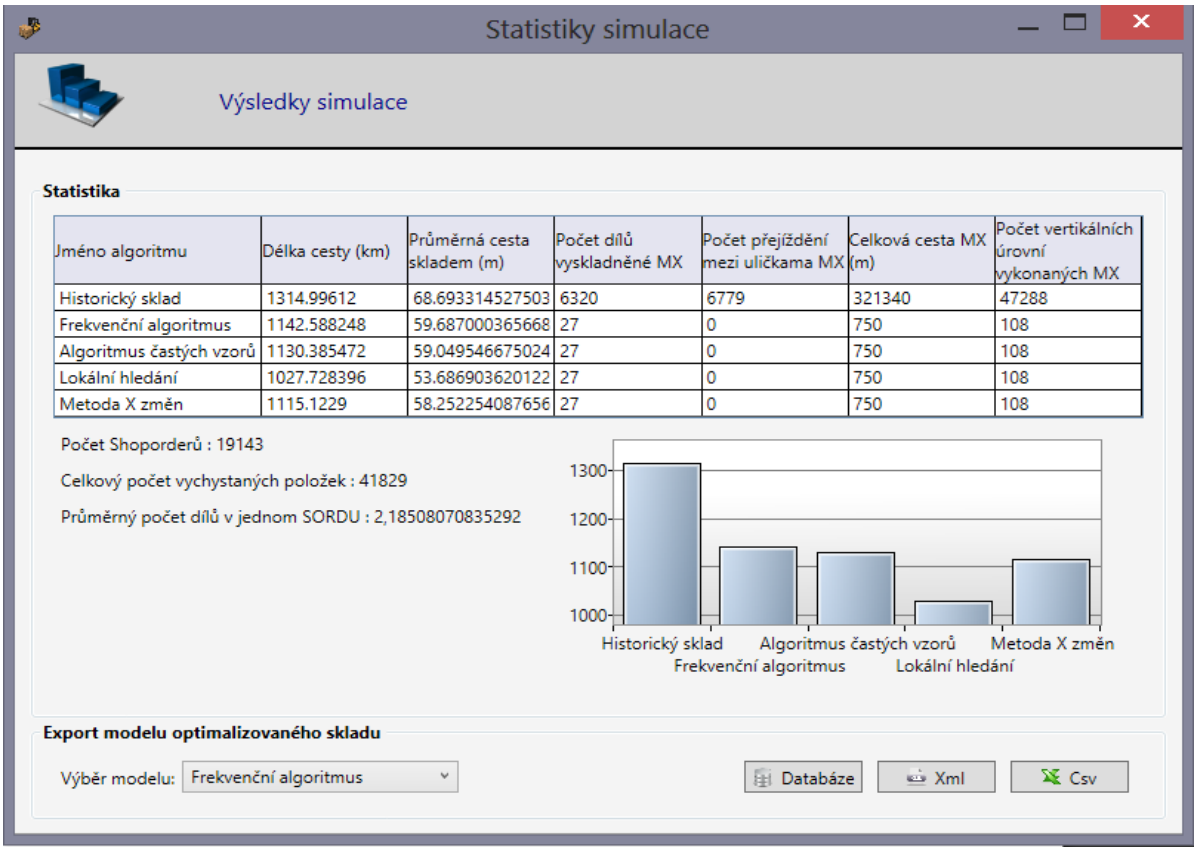
Hlavní okno bude rozděleno do dvou částí. První část (implicitní) bude určena pro optimalizaci skladu. Zde si uživatel nejprve vybere zdroj a rozsah historických dat, jež budou mít vliv na optimalizaci. Druhým krokem bude výběr algoritmu, dle něhož se bude hledat lépe optimalizovaný model a nastavení jeho parametrů. Uživatel zde může zvolit i více algoritmů najednou. Posledním krokem bude již spuštění optimalizace. Zde bude mít uživatel možnost sledovat průběh, případně optimalizaci zastavit. Po dokončení optimalizace je automaticky spuštěna i simulace, aby uživatel mohl zhodnotit přínos simulace. Zjednodušené návrhy uživatelského rozraní jsou zobrazeny na obrázku 4.6.



Obrázek 4.6: Zjednodušený návrh grafického uživatelského rozhraní (*mockup*)

Druhá část zahrnuje simulaci již vytvořeného modelu skladu. Zde se postup liší pouze tím, že místo výběru algoritmu uživatel vybírá uložený model skladu určený pro simulaci. Uživatel může zvolit model z databáze nebo ze souborů CSV a XML. Zde bude probíhat pouze simulace, proto uživateli stačí zobrazovat grafický ukazatel průběhu – *„progress bar“*.

Další důležitou částí bude okno zobrazující výsledky simulace. Uživateli se toto okno otevře bezprostředně po skončení simulace. Statistické údaje budou prezentovány formou tabulky. Pokud je simulace prováděna na modelu, který byl vytvořen při optimalizaci, může uživatel tento model exportovat do databáze či souboru.



Obrázek 4.7: Výpis statistik o simulaci ve výsledné aplikaci

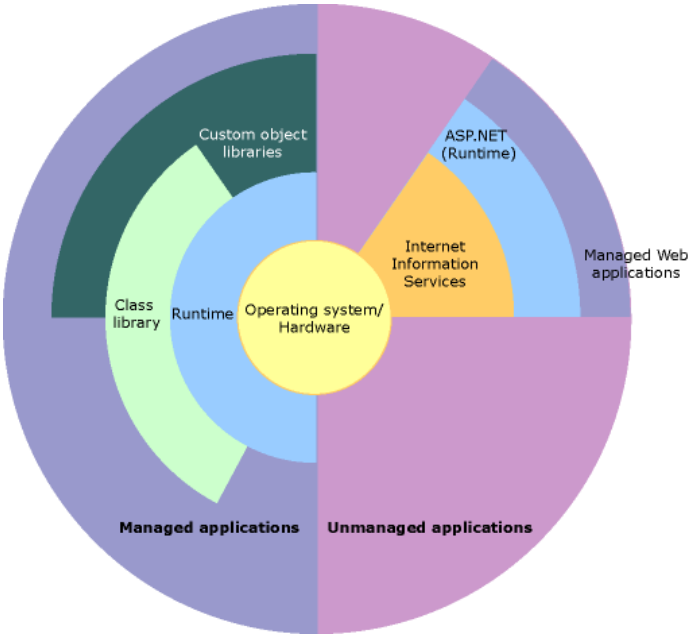
Kapitola 5
Použité technologie

Při implementaci aplikace byl použit jazyk C# a nástroje poskytované firmou Microsoft. Celý systém (včetně skladového) firmy Gates je postaven na těchto technologiích, proto bylo z hlediska udržovatelnosti a konzistence systému vhodné použít právě tuto sadu nástrojů.

5.1 Microsoft .NET framework

Microsoft .NET framework je rozhraní umožňující vytváření a spouštění aplikací nebo webových služeb. Poskytuje konzistentní objektově orientované programovací prostředí. Kód objektu může být uložen a spuštěn lokálně, spuštěn lokálně a distribuován pomocí internetu nebo spouštěn vzdáleně.

Framework nepředepisuje použití konkrétního jazyka a kód se přeloží do mezijazyka *Common intermediate language*, který je následně prováděn modulem *Common language runtime*. Nejčastěji používané jazyky jsou C#, Visual Basic a Delphi. Je možné použít i jazyky jako managed C++, F#, J# (jazyk podobný Javě) a IronPython. V praxi se pro vývoj aplikací s využitím tohoto frameworku nejčastěji používá vývojové prostředí Microsoft Visual Studio. [4]



Obrázek 5.1: Architektura .NET frameworku [4]

5.2 C#

C# je jazyk vyvinutý firmou Microsoft společně s .NET frameworkem. Jedná se o vysokoúrovňový a objektově orientovaný jazyk. Jeho syntaxe je podobná Javě a C++. Jazyk umožňuje hlídat hranice polí, detekovat použití neinicializovaných proměnných a automatickou správu paměti pomocí garbage collectoru. Jeho použití je možné jak v aplikacích pro zařízení se sofistikovanými operačními systémy, tak i pro vestavěné systémy. I přes to, že by programy psané v C# měly být úsporné z hlediska práce s procesorovým časem a pamětí, nedá se výkon srovnávat s programy psanými

v jazyce C nebo v jazyce symbolických adres (assembleru). Ve verzi 3.0 byl do jazyka integrován LINQ (Language Integrated Query), umožňující dotazovaní nad jakýmikoli daty, a také lambda výrazy, jež jsou inspirovány funkcionálními jazyky. [13]

5.3 WPF

Windows presentation foundation je systém pro tvorbu .NET klientských aplikací s vizuálně zajímavým uživatelským rozhraním. S WPF lze vytvářet jak klasické desktopové aplikace, tak i aplikace hostované prohlížečem. Jádro WPF je založeno na vektorových objektech a jejich vykreslování je akcelerováno hardwarem. Tento systém poskytuje vývojářům nástroje pro tvorbu a využívání komponent uživatelského rozhraní, data-binding, 2-D a 3-D grafiku, animace, stylování, dokumenty a multimédia. [6]

5.4 XAML

XAML (Extensible Application Markup Language) je deklarativní značkovací jazyk pro popis vzhledu uživatelského rozhraní. Firma Microsoft tento jazyk založila na XML. Nejčastěji se používá pro tvorbu WPF aplikací. Jeho hlavní výhodou je možnost oddělit práci designéra od práce programátora. Designéři nejčastěji používají editor Microsoft Expression Blend, který je pro tvorbu pomocí jazyka XAML přizpůsoben. [4]

Kapitola 6
Implementace

Tato kapitola popisuje implementaci programu pro optimalizaci skladu. Při implementaci se vycházelo z postupů uvedených v kapitole 4. Aplikace je určena pro operační systém Windows a její nasazení je možné pomocí instalátoru.

6.1 Členění programu

Program je rozdělen do tří projektů Visual Studia. Projekt *StoreSimulation* tvoří jádro simulátoru a optimalizátoru. Tato část zahrnuje hlavní aplikační logiku programu. *StoreModel* obsahuje datový model aplikace. Tento projekt se stará o komunikaci s databází a zajišťuje tak import a export modelů skladu a historických objednávek. Poslední částí je *StoreSimulatorGUI*, jež obsahuje grafické uživatelské rozhraní aplikace.

6.2 Simulátor vychystávání

Simulátor byl navržen a implementován na míru skladu ve firmě Gates Hydraulics. Implementace simulátoru obecného skladu by byla velmi náročná a bylo by nutné vytvořit prostředky pro popis konkrétního skladu. Řešení prezentované v této práci předpokládá sklad s obdélníkovým půdorysem, se stejně velkými lokacemi a s fixními vzdálenostmi mezi regálovými konstrukcemi. Také vychystávací bod pracovníka i startovní pozice vysokozdvižného vozíku jsou pevně určeny. Pro použití aplikace i v jiném skladu by bylo nutné modifikovat parametry skladu (šířky regálů, šířky uliček atd.) a třídu představující skladovou lokaci. Pokud by byl sklad typově jiný (např. neprůchozí uličky), bylo by nutné modifikovat také výpočet nejkratší cesty a způsob pohybu vysokozdvižného vozíku.

6.2.1 Model skladu

Model skladu je reprezentován třídou *Store*. Tato třída obsahuje parametry skladu (vzdálenosti mezi regály, šířky uliček atd.). Hlavním prvkem této třídy je kolekce typu slovník. Klíčem této kolekce je řetězec s názvem dílu a hodnotou typ *Location*, což je třída reprezentující jednu lokaci ve skladě. Tato kolekce umožňuje rychlé vyhledání lokace na základě požadovaného dílu. Vyhledání lokace je klíčová akce při simulaci.

Třída *Store* dále obsahuje metodu *GetRatedLocations()* pro seřazení lokací dle dostupnosti, která je nutná při rozmisťování dílů. Nachází se zde taky metody pro import a export modelu skladu do souborů CSV a XML. Model skladu není ekvivalentem reálného skladu a modeluje pouze sběrovou část skladu. Každému dílu je přiřazena pouze jedna lokace a během simulace se předpokládá, že je díl vždy přítomný. Simulace tedy nijak nezahrnuje proces přesunu materiálu ze zásobní části.

6.2.2 Implementace simulátoru vychystávání

Stěžejní částí simulátoru je třída *Simulator*. Tato třída obsahuje historické objednávky a metody pro spuštění simulace. Simulace se provádí buď s modelem skladu, nebo jsou lokace dílů zjišťovány z historických dat. Pokud jsou lokace zjišťovány z historických dat, délky cest odpovídají drahám, které pracovníci v minulosti opravdu prošli. V případě simulace modelu skladu se pak jedná o dráhy, které by pracovníci ušli, pokud by sklad odpovídal rozložení dílů v modelu skladu. Model skladu

může být i obrazem aktuálního rozložení v reálném skladu. Po skončení simulace jsou vlastnosti instance typu *Simulator* naplněny statistickými údaji o simulaci.

Třída *Simulator* vnitřně používá další komponenty nutné pro běh simulace. Mezi ty nejdůležitější patří třída *PathFinder*, jejíž metoda *FindPath()* na základě kolekce lokací nalezne nejkratší cestu skladem. *PathFinder* využívá výpočet vzdálenosti mezi lokacemi uvedený v kapitole 4.2.1. O rekurzivní generování permutací lokací se zde stará třída *Permutation*, jejíž metoda *FindPerm()* navrací kolekce kolekcí lokací. *Simulator* taktéž obsahuje třídu *Forklift*, která eviduje pohyb vysokozdvižného vozíku v průběhu simulace. Statistické údaje o pohybu vysokozdvižného vozíku jsou pak dostupné skrz vlastnosti třídy *Simulator*.

Simulace na běžném počítači v časovém rozmezí jednoho roku trvá méně než minutu. Nejvíce času stráví algoritmus při hledání optimální cesty. Hledání optimální cesty má faktoriálovou složitost, proto pokud se vyskytne objednávka obsahující větší počet typů dílů, je čas potřebný k simulaci podstatně delší.

6.3 Optimalizace modelu

Výstupem optimalizace by měl být model skladu, který je výhodnější než model aktuální. Toho lze dosáhnout pomocí metod uvedených v kapitole 4.3. Optimalizace tedy může mít více implementací. Třídy, jež reprezentují optimalizaci, implementují rozhraní *IAlgorithm*. Toto rozhraní předepisuje metodu *AssignItems()*. Výstupem metody je optimalizovaný model skladu. K algoritmům lze tedy přistupovat skrz toto rozhraní, aniž bychom znali vnitřní implementaci.

6.3.1 Rozmístění dle popularity

Tato strategie k rozmisťování dílů je nejjednodušší a její implementace se nachází ve třídě *FreqAlgorithm*. Algoritmus používá třídu *ItemsAnalytics*. Třída obsahuje kolekci všech dílů seřazenou dle jejich četností, jež byla vytvořena z historických objednávek. Z této kolekce se vytvoří optimalizovaný model. Tento algoritmus je nejrychlejší a vygenerování modelu skladu trvá na běžném počítači zlomek sekundy.

6.3.2 Algoritmus častých shluků

Třída *FreqPatternAlgorithm* popisuje rozmisťování dle častých shluků. Konstruktor této třídy obsahuje číselný parametr, který nastavuje minimální podporu shluku. Tento algoritmus je mírně pomalejší než předchozí implementace, avšak ne příliš výrazně. Rychlost je závislá na nastaveném parametru minimální podpory. Na běžném počítači výpočet netrvá déle než jednu minutu.

6.3.3 Lokální hledání

Tato heuristická metoda optimalizace je obsažena ve třídě *LocalSearch*. Před samotným hledáním je konstruktorem nastaven počet iterací. Předem jsou také připraveny četnosti dílů a četnosti dvojic dílů, které jsou nutné k rychlému stanovení skóre rozmístění dle vzorce v kapitole 2.4.2. O tento výpočet se starají statické metody třídy *ScoreCounter*. Generování náhodných lokací ve skladu je implementováno ve třídě *RandomLoc*.

Vyvážení součtu skóre dle popularity a skóre dle častosti zboží bylo experimentálně určeno z dosažených výsledků simulace. Rychlost algoritmu záleží na počtu iterací. Pro výraznější optimalizaci je potřeba větší počet iterací a následné hledání trvá i několik minut.

6.3.4 Metoda zadaného počtu změn

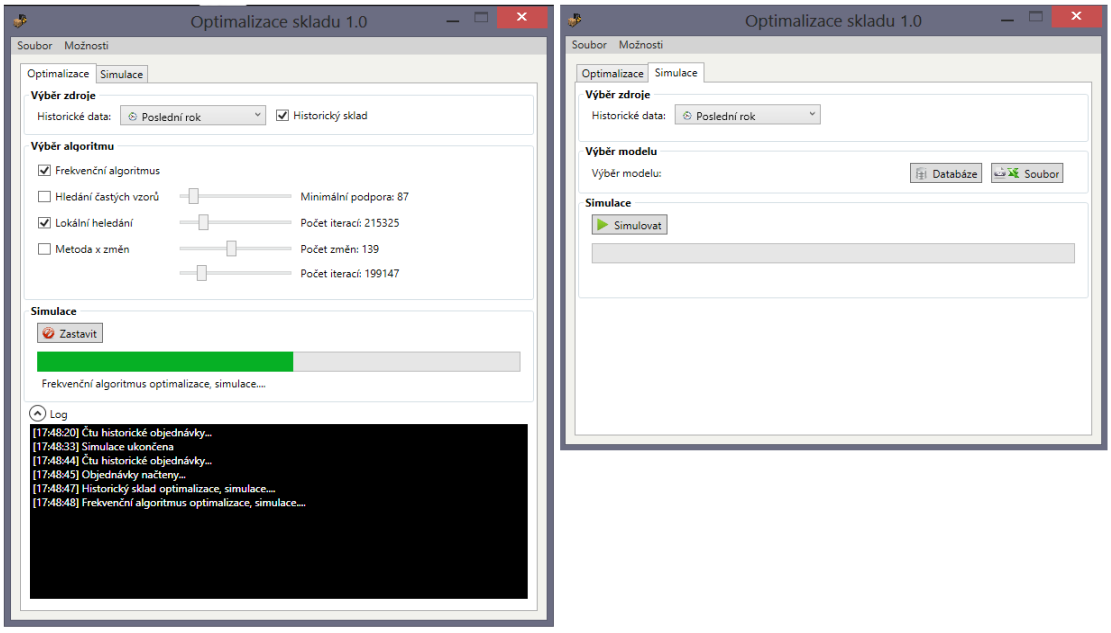
Tento algoritmus se nachází ve třídě *TopSwap*, jež dědí od třídy *LocalSearch*. Oproti svému předkovi vychází z modelu aktuálního skladu a uživatel nastavuje navíc i počet změn, které mají být na modelu provedeny a vedou k lepší optimalizaci rozmístění. Při iteraci jsou změny evidovány v generické kolekci typu *List* a během hledání se zde udržují jen změny s nejpozitivnějším vlivem na kvalitu rozmístění. Rychlost tohoto algoritmu je srovnatelná s lokálním hledáním.

6.4 Implementace uživatelského rozhraní

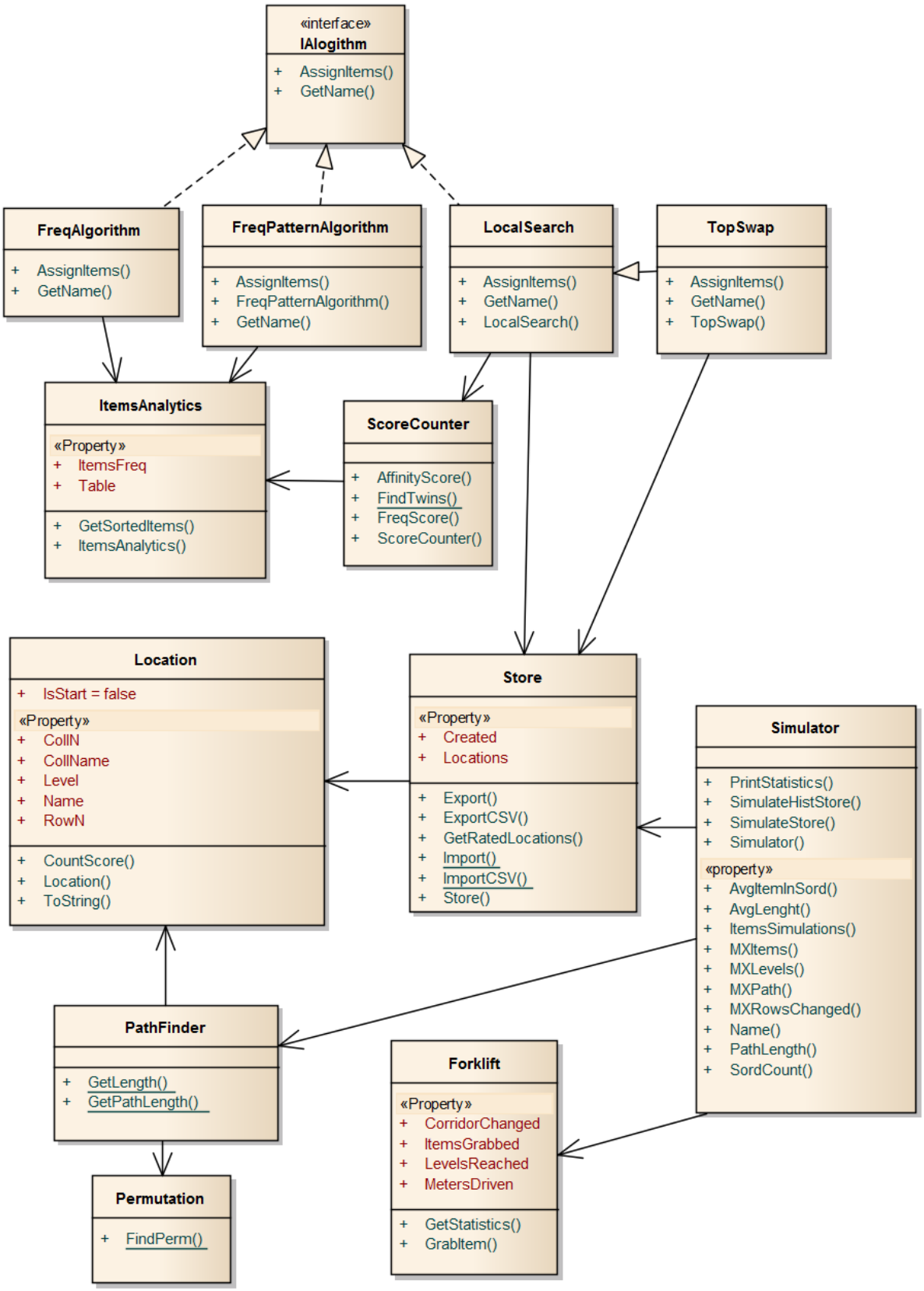
Vzhled uživatelského rozhraní je popsán pomocí jazyka XAML. Hlavní okno aplikace je rozděleno do záložek na optimalizaci a simulaci. Průběh simulace/optimalizace je uživateli prezentován pomocí grafického ukazatele průběhu. Tento prvek je pravidelně aktualizován pomocí asynchronních delegátů. O zaznamenávání průběhu výpočtu se stará třída *Progress*, jež sdružuje stavy jednotlivých časově náročných výpočtů. Před časově náročným výpočtem se činnost zaregistruje pomocí metody *NewTask()*, která vrací identifikační číslo. Tímto číslem se pak identifikuje činnost při volání metody *UpdateProgress()* sloužící k zaznamenání stavu průběhu dané činnosti. Třída *Progress* poskytuje vláknu, jež se stará o aktualizaci grafického ukazatele průběhu, vlastnost *State*. Tato vlastnost nese procentuální stav celého výpočtu. Třída *Progress* je zabezpečena z hlediska současného přístupu více vláken.

Uživatel při optimalizaci může spustit najednou i více metod optimalizace. Jednotlivé úlohy k provedení jsou zapouzdřeny do třídy *Tasks*. Tato třída obsahuje výchozí model skladu, instanci optimalizačního algoritmu, třídu *Simulator* a statistiků dílů. Před samotným výpočtem je vytvořena kolekce těchto úloh a ty jsou pak postupně zpracovávány. Pro zvýšení efektivity výpočtu by bylo možné tyto úlohy zpracovávat paralelně.

Uživatelské rozhraní obsahuje pomocná okna pro import a export do databáze. V sekci nastavení dostupné z menu může uživatel nastavit připojovací řetězec k databázi. Při nasazování této aplikace do firmy bude nutné vytvořit na firemním serveru databázi s příslušnými tabulkami a pohledy. Do konfiguračních souborů instalátoru aplikace se pak vloží připojovací řetězec k této databázi a uživatel bude schopen po instalaci program ihned využívat.



Obrázek 6.1: Snímky záložek hlavního okna aplikace



Obrázek 6.2: Diagram tříd jádra aplikace

Kapitola 7
Testování a dosažené výsledky

V této kapitole budou uvedeny postupy, které byly použity při testování aplikace. Zmíněny zde budou také problémy, které se vyskytly během testování na jiných verzích systému Windows. V kapitole 7.2 pak budou prezentovány výsledky jednotlivých optimalizačních metod.

7.1 Testování aplikace

Z návrhu aplikace je vidět, že mnohé její části mohou být rizikové. Proto byla aplikace vystavena sadě testů, které kontrolovaly reakce programu na neočekávané stavy. Jedná se o případy, kdy uživatel nezadá vstupní soubor, vstupní soubor je nevalidní, připojení k databázi není možné a podobně. Tyto chyby je vhodné uživateli sdělit dříve, než započne samotný výpočet.

Další oblastí možných rizik je samotná simulace modelu skladu, kdy může dojít k tomu, že se v historických datech objeví díl, který se v simulovaném modelu nevyskytuje. Při této chybě bude potřeba simulaci ukončit. Ukládání modelů skladů do databáze či souboru patří také mezi rizikové oblasti a uživateli by mělo být jasně sděleno, proč se daná operace nezdařila.

7.1.1 Testování jádra aplikace

Jádro aplikace obsahuje netriviální výpočty a jejich správnost musí být také ověřena. Nejrizikovější je hledání cesty skladem. Tento problém lze účinně testovat vyhledáváním optimální cesty skladem, u které si dopředu nejkratší cestu vypočítáme ručně.

A				C
	D		B	

Obrázek 7.1: Testování průchodu skladem

Na obrázku 7.1 se nachází čtyři lokace, které skladník musí navštívit. Předpokládejme, že se startovní bod nachází v pravém dolním rohu obrázku. Na první pohled je zde patrné, že nejkratší cesta, při které budou navštíveny všechny lokace, se rovná obvodu regálové konstrukce. Program bude postupně zkoušet všech 24 možností, v jakém pořadí lokace navštívit, a měl by vybrat tu nejkratší variantu. Správnost výpočtu nejkratší cesty byla testována sadou podobně stavěných testů, které obsahovaly i složitější průchody skladem. Samotná simulace pak byla testována na malém počtu objednávek, u kterých bylo možné výslednou statistiku odhadnout či vypočítat bez použití aplikace.

Optimalizátor i jeho samotné algoritmy byly testovány pomocí opakovaného spouštění s různými parametry a časovými intervaly. Sledovány byly zejména výsledky, jakých optimalizace dosahuje. I když uživatelské rozhraní nedovoluje zadat uživateli nevalidní parametry, samotné algoritmy tyto hodnoty kontrolují a při výskytu nevalidních parametrů vyvolají programovou výjimku.

Aplikace byla vyvíjena na operačním systému Windows 8 a následně otestována také na systémech Windows 7 a Windows XP. Při testování na těchto starších systémech se objevily problémy zejména s uživatelským rozhraním aplikace, které byly později odstraněny. Na jednom testovacím stroji se však aplikaci nepodařilo vůbec spustit. Tato chyba byla odstraněna aktualizací .NET frameworku.

7.2 Dosažené výsledky

V této kapitole se nachází výsledky dosažené různými optimalizačními metodami. Prezentována zde bude především celková cesta skladníků za období jednoho roku. Pohyb vysokozdvižného vozíku není tak relevantní, protože při simulaci optimalizovaného skladu nemáme přehled o počtu dílů v dané lokaci. Simulátor tedy předpokládá, že je díl vždy přítomen a nemá možnost započítat pohyb vysokozdvižného vozíku při přesunu dílu ze zásobní části. Pohyb vysokozdvižného vozíku při simulaci optimalizovaného skladu eviduje pouze pohyby pro ručně nedostupné díly. Z tohoto důvodu je tedy pohyb vysokozdvižného vozíku při simulaci optimalizovaného modelu několikanásobně nižší oproti simulaci historického skladu. Hodnocení efektivity rozmístění na základě celkové cesty byl popsán například v [1].

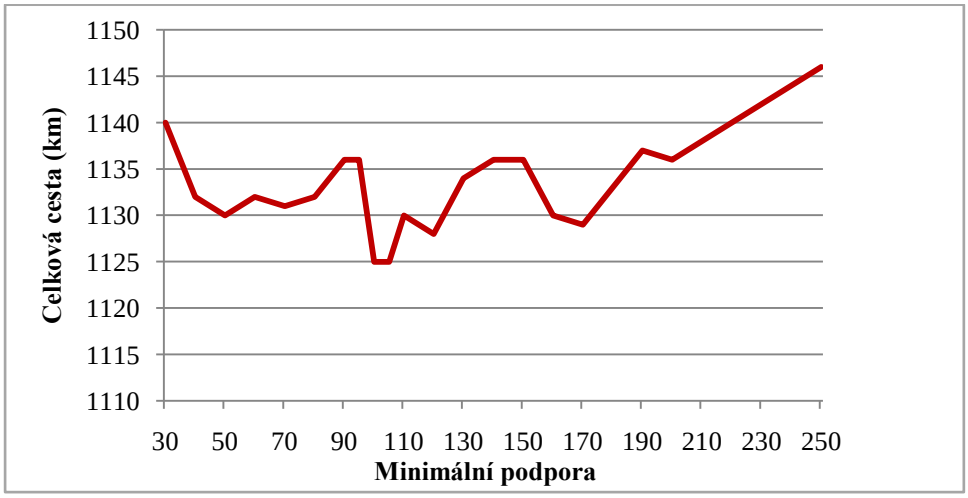
Testovací sada objednávek zahrnovala reálná data objednávek za období jednoho roku. Vylepšení rozmístění je vždy vztahováno k náhodnému uložení dílů. Porovnání vůči historickým údajům by nebylo tak vypovídající, jelikož se nepočítá s pevným uložením dílů a mnoho dílů bylo vyskladněno vysokozdvižným vozíkem, což by ovlivnilo celkovou cestu pracovníků. Celková cesta při náhodném rozmístění dílu činila 1772 km.

7.2.1 Rozmístění dle popularity

Při aplikaci tohoto rozmístění se oproti náhodnému rozmístění snížila výsledná cesta o 36 %. V případě snížení sledovaného časového období z jednoho roku na 3 měsíce pak bylo snížení cesty srovnatelné. Tyto výsledky nasvědčují, že i touto jednoduchou metodou lze aktuální stav znatelně zlepšit.

7.2.2 Algoritmus častých shluků

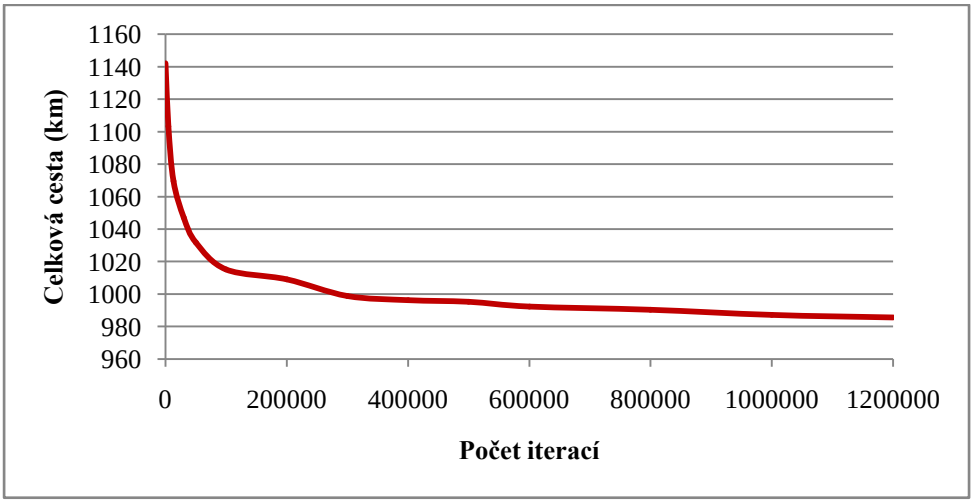
Kvalita optimalizace touto metodou závisela na vstupním parametru minimální podpory. Závislost tohoto parametru na celkové cestě je znázorněna v grafu níže. Jak lze vidět, hledání ideální hodnoty parametru by vyžadovalo řešení dalšího optimalizačního problému, což by bylo vzhledem k časově náročné simulaci velice neefektivní. Metoda však ve většině případů dosahovala mírně lepších výsledků oproti rozmístění dle popularity, kdy byla celková cesta vyčíslena na 1142 km.



Obrázek 7.2: Efektivita modelu v závislosti na minimální podpoře častých shluků

7.2.3 Metoda lokálního hledání

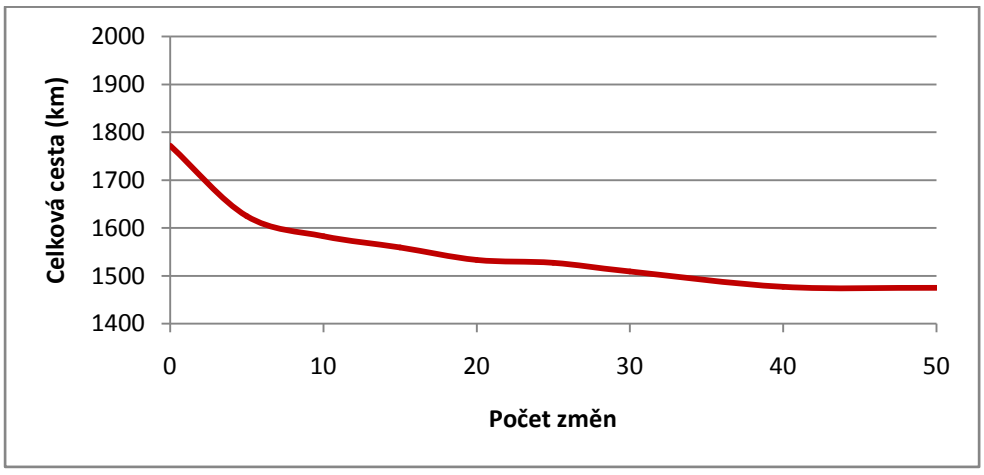
Metoda dosahovala podstatně lepších výsledků než předchozí přístupy. Kvalita rozmístění záležela na počtu provedených iterací. U této metody hrozí uváznutí v lokálním minimu[2], které se však v tomto případě projevovalo jen výjimečně a výsledek nebyl výrazně horší oproti jiným simulačním běhům. Hodnoty na grafu uvedeném níže jsou průměry jednotlivých běhů se stejným počtem iterací. Počátečním modelem skladu bylo rozmístění na základě popularity. Použití metod pro zrychlení hledání, jež jsou navrženy ve třetím odstavci v kapitole 4.3.4, se neosvědčilo a při pokusech aplikovat tyto postupy algoritmus dosahoval horších výsledků.



Obrázek 7.3: Závislost celkové cesty na počtu iterací lokálního hledání

7.2.4 Metoda zadaného počtu změn

Kvalita rozmístění touto metodou podobně jako u lokálního hledání záležela na počtu iterací. Zde se však hodnota mnohem rychleji ustálila. Počátečním modelem bylo náhodné rozmístění skladu a počet iterací byl nastaven na 50.000. Jak lze vidět z grafu, tak i při provedení malého počtu změn lze dosáhnout výrazně nižší celkové cesty. U této metody se však projevovalo uváznutí v lokálním minimu a uvedené hodnoty jsou minimem z více optimalizačních běhů.



Obrázek 7.4: Závislost celkové cesty na počtu provedených změn

[2] Lokálním minimem funkce rozmístění skladu je zde myšleno takové rozmístění, z něhož se jen s velice malou pravděpodobností lze dostat do lepšího stavu pomocí výměny pozic dvou dílů.

Tato metoda je nejvýhodnější z hlediska aplikace rozmístění do reálného skladu. Pokud je zadaný počet změn roven 10, ve skladu se přemístí pouze 20 dílů ve sběrové části skladu. U ostatních metod by bylo nutné přemístit až 900 dílů ve sběrové části.

7.2.5 Shrnutí výsledků použitých metod

V níže uvedené tabulce jsou srovnány jednotlivé metody rozmístění. Uvedené časy byly naměřeny na stroji s procesorem Intel i5-2520M se základní frekvencí 2,5 GHz. U heuristických algoritmů byl nastaven milion iterací a u metody zadaného počtu změn jich bylo provedeno 10.

Použitá metoda	Celková cesta (km)	Ušetřená cesta (km)	Ušetřená cesta za den (km)	Časová náročnost výpočtu (s)
Náhodné rozmístění	1772			
Dle popularity	1142	630	2,65	<1
Častých shluků	1125	647	2,73	1
Lokální hledání	985	787	3,32	599
Zadaného počtu změn	1583	189	0,8	659

Tabulka 7.1: Srovnání efektivity jednotlivých metod rozmístění

Kapitola 8
Závěr

V rámci práce byla vytvořena aplikace, která umožňuje optimalizovat rozmístění dílů ve skladu firmy Gates Hydraulics dle čtyř algoritmů. Žádný z těchto algoritmů se nedá považovat za ideální, ale v praxi se nejspíše nejvíce uplatní metoda zadaného počtu změn. Aplikace je navržena tak, že ji nebude problém rozšířit o nové algoritmy. Uživatelské rozhraní je jednoduché a umožňuje uživateli rychle spustit simulaci či optimalizaci.

I když byla aplikace navržena tak, aby byla výkonná, optimalizaci rozmístění i proces simulace by bylo možné dále optimalizovat. Při vývoji byl použit nástroj pro profilování. Výkonnost simulace se dvojnásobně zvýšila po odstranění redundantních alokací paměti. Pro další zvýšení výkonu je možné implementovat některou z heuristických metod hledání nejkratší cesty. Další možností, jak zvýšit výkon aplikace, by bylo rozdělení výpočtu do více vláken.

Pro uživatele by se dalo dosáhnout určitého vylepšení pokročilejším sledováním stavu výpočtu – jednotlivé operace mají při prezentaci na grafickém ukazateli průběhu stejně velký díl, nehledě na jejich časovou náročnost. Tento přístup způsobuje nepravidelné skoky při grafické prezentaci průběhu, což uživatele mate při odhadu délky trvání výpočtu.

Pro zlepšení efektivnosti rozmístění by bylo vhodné zavedení uživatelské korekce rozmístění. Pokud se předpokládá, že se určitý díl bude rušit, měl by se tento díl při hledání optimalizovaného skladu vyloučit. Naopak pokud by firma byla schopna předpovědět, že se brzy vyskytne nový díl, který bude nutné skladovat, bylo by vhodné určit jeho předpokládanou častost vychystávání a zahrnout ho do rozmístění.

Výsledná aplikace bude v nejbližší době zavedena do systému firmy Gates. Při praktickém využití se mohou vyskytnout určité nedostatky, ale praxe také ukáže další cesty a možnosti vylepšení. Na aplikaci se tedy bude i nadále pracovat ve spolupráci s IT oddělením firmy a vedoucími pracovníky skladu.

Literatura

[1] CHUANG, Yi-Fei, Hsu-Tung LEE and Yi-Chuang LAI. Item-associated cluster assignment model on storage allocation problems. *Computers & Industrial Engineering*. 2012, č. 63, s. 1171–1177.

[2] DE KOSTER, René, Tho LE-DUC and Kees Jan ROODBERGEN. Design and control of warehouse order picking: A literature review. *European journal of operational research.* 2007, č. 182, s. 481-501. ISSN 0377-2217.

[3] DE KOSTER, René and Edo VAN DER POORT. Routing orderpickers in a warehouse: A comparison between optimal and heuristic solutions. *IIE Transactions*. 1998, č. 30,s. 469–480.

[4] Getting Started with the .NET Framework. *Microsoft Developer Network* [online]. 2012 [cit. 2013-04-23]. Dostupné z: http://msdn.microsoft.com/en-us/library/vstudio/hh425099

[5] HENN, Sebastian, Sören KOCH and Gerhard WÄCSCHER. *Order batching in order picking warhouses*. Magdeburg, 2011. Working paper series. Otto von Guericke Universität Magdenburg.

[6] Introduction to WPF. *Microsoft Developer Network* [online]. 2012 [cit. 2013-04-23]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa970268.aspx

[7] KALLINA, Carl and Jeffrey LYNN. Aplication of the cube-per-oder index rule for stock locations in distribution warehouse. *Interfaces* 1976. č. 7.

[8] KOFLER, Monika and Michael AFFENZELLER. Warehouse Storage Assignment Optimization in an Order Picking Environment. *DOMINIO* [online]. 2004. [cit. 2013-03-28]. Dostupné z: http://domino.uni-graz.at/IPL-Extern/doc.nsf/493ed380d72822d8c12575be003107be/d65c0f0eccb6f154c12577e70030ca3d/$FILE/Kofler%20M.pdf

[9] LAMBERT, Douglas. *Logistika*. 1. vyd. Praha: Computer Press, 2000, 589 s. ISBN 80-722-6221-1.

[10] NAGEL, Christian. *C# 2008: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2009, 772 s. ISBN 978-80-251-2401-7.

[11] ROODBERGEN Kees Jan and René DE KOSTER. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*. 2001, ročník 39, č. 9, s. 1865-1883.

[12] SEDGEWICK, Robert. Permutation generation methods. *Computer Science Department at Princeton University* [online]. 2013 [cit. 2013-05-02]. Dostupné z: http://www.cs.princeton.edu/~rs/talks/perms.pdf

[13] Standart ECMA-334 C# Language Specification. *Welcome to Ecma International* [online]. 2006 [cit. 2013-04-20].
Dostupné z: http://www.ecma-international.org/publications/standards/Ecma-334.htm

[14] VANĚČEK, Drahoš. *Logistika*. 3. přeprac. vyd. České Budějovice: Jihočeská univerzita, 2008, 178 s. ISBN 978-807-3940-850.

[15] ZENDULKA, Jaroslav aj. *Získávání znalostí z databází*. Studijní opora. Brno, 2009.

Seznam příloh

Příloha A Manuál aplikace

Požadavky:

- operační systém Windows XP a vyšší
- .NET framework 4.0 client profile (stažen instalátorem)
- Microsoft Excel 2007 a vyšší
- připojení k internetu

Instalace:

1. Ve složce */dist* spusťte soubor *setup.exe*.
2. Klikněte na tlačítko *instalovat.*
3. Vyčkejte, dokud instalace neskončí. Instalátor se postará o stažení potřebných souborů z internetu.
4. Na ploše vám instalátor vytvoří zástupce *Optimalizace skladu.*

Použití bez připojení k databázi:

1. V části *výběr zdroje* vyberte historická data pomocí položky *Import Excel* a zvolte soubor s historickými daty. Testovací data se nacházejí ve složce *hist_data* na přiloženém médiu.
2. Zaškrtněte políčko *Historický sklad.* Model aktuálního rozložení skladu není k dispozici.
3. Vyberte, které metody chcete použít, a nastavte jejich parametry. V případě použití programu bez připojení k databázi bude metoda zadaného počtu změn vycházet z rozmístění dle popularity.
4. Stiskněte tlačítko simulovat a vyčkejte, až simulace skončí.
5. Prohlédněte si statistiku. Vybrané optimalizované modely si můžete uložit do formátu CSV nebo XML.

Při simulaci postupujte obdobným způsobem. Model skladu vyberte ze souboru.

Připojení k databázi:

1. V horním pruhu menu zvolte *Možnosti – Nastavení* a vložte připojovací řetězec k databázi.
2. Potvrďte tlačítkem *Ok*.

Příloha B Obsah CD

Na přiloženém CD se nacházejí následující adresáře a soubory:

- **/src** – zdrojové soubory s projektem Visual Studia
- **/dist** – instalační balíček aplikace
- **/BP-Optimalizace_skladu.pdf** – bakalářská práce ve formátu PDF
- **/hist_data** – historická data ve formátu XLSX pro testování
- **/models** – uložené modely skladů
- **/manual.pdf** – manuál aplikace ve formátu PDF