České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Rozvrhování služeb v nemocnici

Praha, 2007 **Autor: Zdeněk Bäumelt**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 19. 1. 2007

Bäuerle

podpis

Poděkování

V úvodu bych rád poděkoval zejména vedoucímu Ing. Přemyslu Šůchovi a jeho kolegům za odborné vedení, cenné rady a připomínky během tvorby této diplomové práce. Dále bych chtěl poděkovat svým rodičům a blízkým za veškerou podporu, které se mi v průběhu studia nejen na vysoké škole dostalo. Děkuji i všem ostatním, kteří mi jiným způsobem pomohli při tvorbě této práce.

Abstrakt

Cílem této diplomové práce je návrh aplikace pro řešení problému rozvrhování služeb ve zdravotnických zařízeních (*Nurse Scheduling Problem*). Jak již název napovídá, jedná se o problém z oblasti rozvrhování (*scheduling*, *timetabling*). Během řešení tohoto NP-obtížného problému je nezbytné uvažovat velké množství různých omezení (minimální počet sester na směně, pracovní vytíženost sester, rovnoměrné obsazení směn, požadavky sester na rozvrh, atd.). Dalším nezanedbatelným aplikačním požadavkem je navrhnout rozvrh v přijatelném čase.

Náplní této práce je seznámit čtenáře s různými rozvrhovacími algoritmy vhodnými pro řešení a implementovat vlastní rozvrhovací algoritmus. Výsledkem je webová aplikace pro rozvrhování služeb v nemocnicích navržená v prostředí .NET.

Abstract

The aim of this diploma thesis is the development of an application for solving *Nurse Scheduling Problem*. This problem belongs to *scheduling problems* domain, exactly *timetabling problems* domain. For solving this NP-hard problem, it is necessary to consider large amount of constraints (minimum number of nurses for each type of shift, nurses' workload, balanced shift assigning, nurses' requests consideration, etc.). Another application constraint is designing schedule in an acceptable period of time.

The scope of this work is to introduce various scheduling algorithms suitable for solving the problem and to implement a self-made scheduling algorithm. The final product of this diploma thesis is a web application for solving Nurse Scheduling Problem implemented in .NET Framework.

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Zdeněk Bäumelt

Obor: Technická kybernetika

Název tématu: Rozvrhování služeb v nemocnici

Zásady pro vypracování:

1. Seznamte se s problematikou rozvrhování služeb v nemocnicích.
2. Navrhněte řešení problému.
3. Implementujte a odzkoušejte navržené řešení.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Přemysl Šůcha

Termín zadání diplomové práce: zimní semestr 2004/2005 (změna zadání: 20.02.2006)

Termín odevzdání diplomové práce: leden 2007

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze dne 27. 02. 2006

Obsah

Seznam obrázků **x**

Seznam tabulek **xi**

Seznam použitých zkratek **xii**

Seznam použitých proměnných **xiv**

1 Úvod **1**

1.1 Definice problému 2

1.2 Související práce 2

1.2.1 Optimalizační metody (matematické programování) 2

1.2.1.1 Lineární a celočíselné programování 3

1.2.1.2 Cílové programování 3

1.2.2 Metody umělé inteligence 4

1.2.2.1 Deklarativní programování 4

1.2.2.2 Expertní systémy 4

1.2.3 Heuristické metody 5

1.2.3.1 Tabu Search 5

1.2.3.2 Genetické algoritmy 5

1.2.4 Implementované nástroje pro NSP 6

1.3 Osnova práce 6

2 Nalezení optimálního řešení **8**

2.1 Syntaxe a programování 8

2.1.1 Syntaxe 8

2.1.2 Obecná definice problému LP v GLPK 10

2.2 Definice problému NSP 11

2.2.1 Slovní formulace omezení 11
2.2.2 Matematický model 12
2.2.2.1 Použité proměnné v matematickém modelu 13
2.2.2.2 Formulace nutných omezení 13
2.2.2.3 Formulace doplňujících omezení 14
2.2.2.4 Stanovení cílové funkce 15
2.3 Výsledky 16
2.3.1 Výsledky pro matematický model se třemi doplňujícími omezeními 17
2.4 Zhodnocení 20

3 Nalezení heuristického řešení 23
3.1 Úvod 23
3.2 Matematický model heuristického řešení 23
3.2.1 Formulace nutných omezení 24
3.2.2 Formulace doplňujících omezení 25
3.2.3 Stanovení cílové funkce Z 25
3.3 Návrh algoritmu 26
3.3.1 Vstupní blok algoritmu 27
3.3.2 Inicializační blok algoritmu 27
3.3.3 Optimalizační blok algoritmu 28
3.3.3.1 Metaheuristika Tabu Search 29
3.3.3.2 Aplikace Tabu Search v rozvrhovacím algoritmu 31
3.3.3.3 Ukončovací podmínka optimalizačního bloku 32
3.3.3.4 Parametry optimalizačního bloku 33
3.3.4 Výstupní blok algoritmu 33
3.4 Pseudokód algoritmu heuristického řešení 34
3.5 Výsledky 40
3.6 Zhodnocení 43

4 Specifikace aplikace 44
4.1 Specifikace grafického rozhraní aplikace 45
4.1.1 Základní myšlenky návrhu GUI 45
4.1.2 Specifikace GUI pro řadového uživatele (sestru) 46
4.1.3 Specifikace GUI pro administrátora 48
4.2 Návrh struktury aplikace 52

4.2.1 Návrh databáze, E-R diagram 53
4.2.2 Objektově orientovaná analýza aplikace 54
4.2.3 Strukturovaná analýza aplikace 57
4.3 Zhodnocení 58

5 Implementační část **59**
5.1 Struktura aplikace 59
5.2 Grafické uživatelské rozhraní aplikace 59
5.3 Rozvrhovací algoritmus 63
5.4 Struktura návrhu rozvrhu 65
5.5 Budoucí zlepšení 66
5.6 Experimentální výsledky 67
5.7 Zhodnocení 69

6 Závěr **70**

A Obsah přiloženého CD **71**

Literatura **74**

Seznam obrázků

1.1 Metodika práce. 7

2.1 Teoretická závislost výpočetního času rozvrhu na počtu sester. 21
2.2 Závislost výpočetního času rozvrhu na počtu sester získaná z testů. . . . 22

3.1 Blokové schéma algoritmu. 27
3.2 Tvorba počátečního přípustného rozvrhu. 28
3.3 Optimalizace na základě vertikálních změn směn mezi sestrami. 29
3.4 Konstrukce ukončovací podmínky. 32
3.5 Funkce heuristického řešení. 39
3.6 Optimalizace Z, příklad 1. 40
3.7 Výsledný rozvrh, příklad 1. 41
3.8 Optimalizace Z, příklad 2. 41
3.9 Výsledný rozvrh, příklad 2. 41
3.10 Optimalizace Z, příklad 3. 42
3.11 Výsledný rozvrh, příklad 3. 42
3.12 Optimalizace Z, příklad 4. 42
3.13 Výsledný rozvrh, příklad 4. 43

4.1 Blokové schéma návrhu aplikace. 44
4.2 Přihlášení do aplikace. 46
4.3 Rozvrh sester na aktuální měsíc. 46
4.4 Editace požadavků na příští měsíc. 47
4.5 Nápověda pro sestry. 47
4.6 Změna hesla uživatele. 48
4.7 Odhlášení z aplikace. 48
4.8 Automatický návrh rozvrhu. 49
4.9 Ruční změny v automatickém návrhu. 49

4.10 Nastavení údajů oddělení. 50
4.11 Databáze zaměstnanců. 50
4.12 Editace v databázi zaměstnanců. 51
4.13 Parametry automatického návrhu. 51
4.14 Nastavení databáze 52
4.15 Struktura aplikace. 52
4.16 E-R diagram. 53
4.17 Diagram užití (Use Case Diagram). 56
4.18 Diagram datových toků (Data Flow Diagram). 58

5.1 Menu aplikace před přihlášením uživatele. 60
5.2 Menu aplikace po přihlášení uživatele. 60
5.3 Nastavení parametrů automatického návrhu. 62
5.4 Vstup požadavků do automatického návrhu. 62
5.5 Výstup z automatického návrhu. 63
5.6 Uložený rozvrh z automatického návrhu. 63
5.7 Metodika návrhu rozvrhu. 66

A.1 Obsah přiloženého cd. 71

Seznam tabulek

2.1 Příkazy v GLPK. 10
2.2 Použité symboly u LP. 11
2.3 Použité proměnné v matematickém modelu pro GLPK. 13
2.4 Použité proměnné v matematickém modelu pro GLPK (2). 14
2.5 Váhy pro penalizaci v matematickém modelu. 16
2.6 Výsledky z GLPK. 17
2.7 Výsledky z GLPK (2). 17

3.1 Penalizace matematickém modelu pro heuristického řešení. 24
3.2 Váhy pro penalizaci v matematickém modelu. 26
3.3 Vstupní parametry algoritmu heuristického řešení. 27
3.4 Vlastnosti proměnných *Tabu List* a *Swap List*. 31

4.1 Význam použitých výrazů v E-R diagramu, datové typy. 54
4.2 Datové položky v E-R diagramu. 55

5.1 Symboly použité při testech. 68
5.2 Testy aplikace (rozvrhovacího algoritmu). 68

Seznam použitých zkratek

Zkratka	Význam
NSP	Nurse Scheduling Problem, optimalizační problém rozvrhování služeb v nemocnici
GLPK	GNU Linear Programming Kit, nástroj pro řešení problému lineárního programování
LP	Linear Programming, lineární programování
MIP	Mixed Integer Programming, smíšené celočíselné programování
ILP	Integer Linear Programming, celočíselné lineární programování
GUI	Graphic User Interface, grafické rozhraní uživatele
UML	Unified Modeling Language, jazyk pro modelování a analýzu systémů
SSADM	Structured Systems Analysis and Design Method, metoda strukturované analýzy

Seznam použitých proměnných

Proměnná	**Význam**
hard constraints	nutná omezení, musí být vždy splněna
soft constraints	doplňující omezení, na těchto omezeních se provádí optimalizace
n	počet dní v rozvrhu
i	index pro dny, $i = 1, 2, ..., n$
m	celkový počet sester pro tvorbu rozvrhu
k	index pro sestry, $k = 1, 2, ..., m$
D_i	požadovaný minimální počet sester pro denní směnu dne i
N_i	požadovaný minimální počet sester pro noční směnu dne i
$XD_{i,k}$	denní směna sestry k dne i, $\forall i, \forall k$
$XN_{i,k}$	noční směna sestry k dne i, $\forall i, \forall k$
$XR_{i,k}$	volno sestry k dne i, $\forall i, \forall k$
d	odchylky u doplňujících omezení, důležité pro penalizaci
Z	cílová (účelová) funkce
ΔZ	zlepšení Z oproti Z_0 po ukončení optimalizace
Z_0	počáteční hodnota Z po provedení inicializace
j	index pro složky funkce Z
p_j	penalizační váhy pro složky funkce Z
Z_j	složky funkce Z
dim_j	dimenze složky Z_j
t_s	scheduling time, doba rozvrhování, rozvrhovací čas
$XE_{i,k}$	ranní směna sestry k dne i, $\forall i, \forall k$
$XL_{i,k}$	odpolední směna sestry k dne i, $\forall i, \forall k$
$XN_{i,k}$	noční směna sestry k dne i, $\forall i, \forall k$
$XR_{i,k}$	volno sestry k dne i, $\forall i, \forall k$
RE	požadovaný minimální počet sester pro ranní směny

Proměnná	**Význam**
RL	požadovaný minimální počet sester pro odpolední směny
RN	požadovaný minimální počet sester pro noční směny
Z_{cand}	kandidát pro změnu v rozvrhu
Z_{next}	hodnota Z v příštím kroku, hodnota Z následníka
Swap List	pole pro ukládání přesunů směn mezi sestrami
Tabu List	pole pro ukládání tabuizovaných přesunů
tabuMemory	počet řádků matice *Tabu List*, konstanta zapomínání
nurse_lack	příznak pro nedostatek sester

Kapitola 1

Úvod

S nezadržitelným nástupem výpočetní techniky v posledních dvou desetiletích téměř ve všech odvětvích se značně změnil náš život a dalo by se říci, že i jeho tempo. Výpočetní techniku potkáváme dnes již na každém kroku, aniž bychom si uvědomili, že ještě před pár lety jsme si museli vystačit s vlastními silami a nedokonalými a zdlouhavými postupy, které vedly ke kýženému cíli.

Výpočetní technika se promítá i do odvětví jako je zdravotnictví, a to v nemalé míře. Smyslem této diplomové práce je usnadnění práce zdravotním sestrám, které mají za úkol navrhovat rozvržení směn pro ostatní sestry z oddělení. Problém návrhu rozvrhu na příští měsíc pro všechny sestry z oddělení tak, aby bylo rozmístění směn „spravedlivé“ a dle požadavků, obnášel v minulosti pro hlavní sestru víkend strávený se sešitem, obyčejnou tužkou, gumou a kalkulátorem v ruce.

Cílem bylo navržení aplikace k rozvrhování služeb v nemocnicích, která by byla pro sestry pohodlná, srozumitelná a přehledná. Přínosem aplikace je zkvalitnění služeb ve zdravotnictví v podobě úspory času a pracovního úsilí hlavním sestrám, které mají za úkol rozvrhy navrhovat. Dalšími výhodami jsou přístupnost k rozvrhu (jedná se o webovou aplikaci), možnost zadávání požadavků na směny od sester přímo do aplikace, variabilita návrhu rozvrhu dle nároků od hlavní sestry. V budoucnosti jsou plánovány i další výhody, jako zasílání krátkých textových zpráv před směnou jako forma upozornění či propojení se mzdovou částí zdravotnictví, kdy data z aplikace budou zdrojem pro tvorbu pracovních výkazů.

Jedná se o aplikaci s ryze praktickým uplatněním, produkt je a bude dále vyvíjen ve

spolupráci s Psychiatrickou léčebnou Bohnice, po osvědčení v této lokalitě bude následovat snaha o rozšíření a adaptaci v dalších zdravotnických zařízeních.

1.1 Definice problému

Rozvrhování služeb v nemocnicích je NP-obtížný optimalizační problém, který spadá do domény rozvrhování (*scheduling* – [1], [2]), přesněji řečeno časového plánování (*timetabling* – [12]). Ještě konkrétněji jej můžeme zařadit do skupiny problémů rozvrhování lidských zdrojů (*personnel scheduling problems* – [5], [17]). S přibývající délkou navrhovaného období a počtem sester dochází ke kombinatorické explozi. Proto je rozumné pokusit se tento problém řešit přibližnými metodami, které bývají často uspokojivé. Cesta k získání řešení, která je kratší, je vykoupena rozvrhem, který nejčastěji není rozvrhem optimálním. Tyto přibližné metody nazýváme *heuristickými* řešeními.

1.2 Související práce

K řešení problému rozvrhování služeb v nemocnicích se dá přistupovat z několika možných pohledů. Velké množství referencí na literaturu je zpracováno v rešerši [1], kde autor uvedl přehled příspěvků k tématu od šedesátých let až do roku 1994. Podobně se tématem zabývá [2], kde je i ukázka, jak při řešení postupovat. Následující odstavce se zabývají klasifikací řešení problému rozvrhování služeb.

1.2.1 Optimalizační metody (matematické programování)

Optimalizace je matematická disciplína, ve které hledáme minimum (resp. maximum) dané funkce $f(x)$ na dané množině M. Tato funkce se nazývá *účelová* či *cílová*. Množina (nazývá se *množina přípustných řešení*) bývá typicky popsána nějakými omezeními (soustavou rovnic, nerovnic apod.). Matematické programování je optimalizace na konečněrozměrných prostorech, kdy množina M je popsána konečným systémem omezení.

1.2.1.1 Lineární a celočíselné programování

Řešení metodou lineárního nebo celočíselného programování ([3]) patří k nejstarším způsobům řešení. Tato technika je popsána v [4], kde je rozvrhování služeb izolováno od problému stanovení počtu sester na oddělení a samotné rozvrhování řešeno technikami matematického (stochastického) programování. Tyto techniky rozdělují rozvrhování do tří částí: dodržování požadavků (kontrola požadavků ze strany nemocnice i ze strany sester), plánování počtu sester (přijetí, propuštění, zaučování, přesunutí do jiného oddělení) a rozvrhování dostupných sester na základě podmínek vzniklých v prvních dvou fázích. Rozvrhování se rozděluje na iterativní a neiterativní část.

Další způsob řešení je popsán ve [5], kde autor přistoupil k problému z hlediska smíšeného celočíselného kvadratického programování (MIP). Práce se zabývá výpočtem potřebného počtu sester určité kvalifikace k pokrytí určitého počtu směn, které jsou klasicky tři osmihodinové nepřekrývající se. Cílová funkce usiluje o minimalizaci rozdílu mezi minimálním počtem sester a počtem sester v oddělení s ohledem na stanovené podmínky. Při rozvrhování algoritmus nezahrnuje osobní požadavky sester, sestry jsou anonymní. I přes tento nedostatek je rozvrh poměrně kvalitní a může sloužit jako vstup do jiného algoritmu, který osobní požadavky bere v potaz.

Při řešení v literatuře [6] byla využita metoda Branch & Bound (metoda větví a mezí), rozvrhování bylo omezené maximálně třísměnným provozem. Algoritmus však zvládal pouze problémy s menší složitostí, což je obecná nevýhoda metody větví a mezí.

1.2.1.2 Cílové programování

Menší část přístupů je prováděna na principu *cílového* programování. Pro dosažení výsledného rozvrhu definuje cílovou úroveň každého kritéria a priority mezi těmito kritérii. Tyto metody zvládají dosáhnout řešení tak rychle, jak je možné dosáhnout cílových úrovní jednotlivých kritérií dle daných priorit. Metoda cílového programování bývá také nazývána jako *vícekriteriální*.

Řešení cílovým programováním je zastoupeno v [7], kde autoři navrhli dvoufázové cílové programovací heuristiky. Práce je zaměřena na minimalizaci počtu sester, minimalizaci počtu sester s nepřípustnými požadavky a minimalizaci odchylky mezi rozvrženým a žádaným (očekávaným od sester) počtem směn (důraz na spokojenost zaměstnanců).

Délka rozvrhovacího období je jeden týden.

Různorodost problému při rozvrhování služeb v nemocnicích ilustruje [8], kdy byl vyvinut vícekriteriální lineární celočíselný algoritmus, který sloužil k řešení problému, kdy sestry jsou zaměstnány ve více nemocnicích. Algoritmus pracuje se sestrami různých kvalifikací, minimalizuje cenu za přesun sester z jedné nemocnice do druhé, maximalizuje kvalitu služeb s ohledem na požadavky sester.

1.2.2 Metody umělé inteligence

Pro řešení se dají využít i některé metody z oblasti umělé inteligence.

1.2.2.1 Deklarativní programování

Jednu skupinu řešení zastupuje deklarativní programování. Deklarativní programování je postaveno na paradigmatu, podle něhož je program založen na tom, co se počítá a ne jak se to počítá. Je zde deklarován vstup a výstup a celý program je chápán jako funkce vyhodnocující vstupy podávající jediný výstup. Podmnožinou deklarativního programování je programování logické (např. rozvrhování pomocí Prologu – [9], [10]). Řešení pomocí deklarativního programování se věnují i další práce ([11], [12]). Přístup pomocí deklarativního programování však naráží na omezení při řešení problémů s větší složitostí.

1.2.2.2 Expertní systémy

Druhou skupinou jsou řešení, ke kterým se využívají expertní systémy. Expertní systém má za úkol poskytovat expertní rady, rozhodnutí nebo doporučit řešení v konkrétní situaci. Expertní systémy jsou navrženy tak, aby mohly zpracovávat nenumerické a neurčité informace a řešit tak úlohy, které nejsou řešitelné tradičními algoritmickými postupy. Expertní systém má dvě základní komponenty, které jsou na sobě relativně nezávislé. Řídící mechanismus pro odvozování závěrů a bázi znalostí.

V případě návrhu rozvrhu pomocí expertních systémů v [13] a [14] má uživatel možnost nastavení jednotlivých vah určitým cílům. Při návrhu tak interaktivně ovlivňuje jeho průběh právě změnou vah.

1.2.3 Heuristické metody

Pro nalezení optimálního řešení je třeba projít kvazi-nepočítatelně mnoho rozvrhů, které vedou k tomu optimálnímu. Nalezení optimálního rozvrhu není možné v polynomiálním čase (problém je NP-obtížný). V praxi se proto místo nalezení optimálního řešení snažíme nalézt řešení, které se tomu optimálnímu bude co nejvíce blížit, ale jeho nalezení nebude tolik časově náročné. Takové řešení nazýváme heuristické.

1.2.3.1 Tabu Search

Jednou z heuristických metod je metoda Tabu Search. Princip metody je popsán při návrhu heuristického řešení v kapitole 3 (konkrétně algoritmus 1). Heuristika Tabu Search byla využita pro řešení NSP v [15], [16] či [17].

1.2.3.2 Genetické algoritmy

Mezi heuristické metody patří také genetické algoritmy. Genetický algoritmus je heuristický postup, který se snaží aplikací principů evoluční biologie nalézt řešení složitých problémů, pro které neexistuje použitelný exaktní algoritmus. Genetické algoritmy, resp. všechny postupy patřící mezi tzv. evoluční algoritmy, používají techniky napodobující evoluční procesy známé z biologie (dědičnost, mutace, přirozený výběr a křížení) pro „šlechtění“ řešení zadané úlohy.

Princip práce genetického algoritmu je postupná tvorba generací různých řešení daného problému. Při řešení se uchovává tzv. populace, jejíž každý jedinec představuje jedno řešení daného problému. Jak populace probíhá evolucí, řešení se zlepšují. Tradičně je řešení reprezentováno binárními čísly, řetězci nul a jedniček, nicméně používají se i jiné reprezentace (strom, pole, matice). Typicky je na začátku simulace (v první generaci) populace složena z naprosto náhodných členů. V přechodu do nové generace je pro každého jedince spočtena tzv. *fitness* funkce, která vyjadřuje kvalitu řešení reprezentovaného tímto jedincem. Podle této kvality jsou stochasticky vybráni jedinci, kteří jsou modifikováni (pomocí mutací a křížení), čímž vznikne nová populace. Tento postup se iterativně opakuje, čímž se kvalita řešení v populaci postupně vylepšuje. Algoritmus se obvykle zastaví při dosažení postačující kvality řešení, případně po předem dané době.

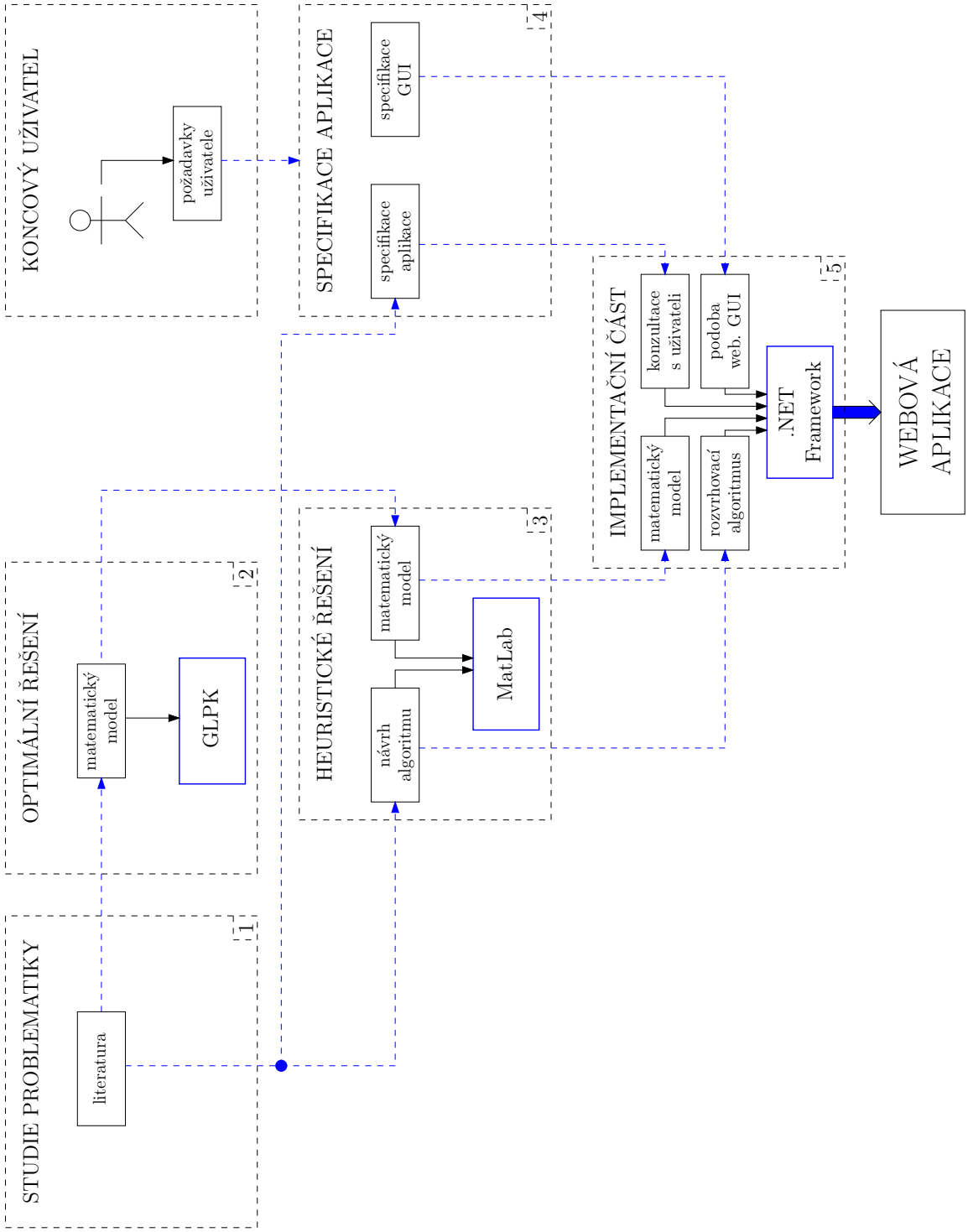
Řešením problému NSP genetickými algoritmy se zabývají např. články [18] a [19].

1.2.4 Implementované nástroje pro NSP

Pro specifikaci aplikace bylo vhodné hledat již implementované volně dostupné nástroje pro rozvrhování služeb v nemocnicích, jejich počet je však velmi nízký. Mezi jeden z nich patří demo produkt společnosti Harborsoft ([20]), kde bylo rozvrhování omezeno počtem sester (max. 25 sester) a rozvrhy bylo možné tvořit do 31. 12. 2006. K instalaci dema bylo potřebné heslo, které bylo k vyžádání zasláním e-mailové zprávy společnosti Harborsoft. Přes veškeré úsilí nejevila společnost snahu odpovědět. Další produkt pochází z Univerzity v Ghentu, fakulty ekonomické ([21]). Bohužel ani tento produkt nebyl při specifikaci přínosem. Třetím produktem byl Time Tracker od společnosti Asgard Systems ([22]), který však není primárně určen pro zdravotnická zařízení a navíc se zdál být poměrně nepřehledným. Při specifikaci bylo nakonec nejvíce čerpáno z nástroje, který v současné době využívají v Psychiatrické léčebně Bohnice ([23]). Tento již zastaralý produkt pracuje pouze v textovém módu (pro specifikaci GUI bezcenný), ale po stránce funkčnosti je kvalitní. Jeho nevýhodou je však nepřehlednost a složitost ovládání, na což byl brán při návrhu aplikace zřetel.

1.3 Osnova práce

Následující odstavce jsou věnovány tématu, jak byla aplikace vyvíjena a jak tomu odpovídá struktura diplomové práce (obr. 1.1, čísla v rozích jednotlivých bloků odpovídají číslům kapitol). Po studii problematiky bylo záměrem nalézt optimální řešení. Čtenář se zde může přesvědčit o ne příliš použitelné formě řešení po aplikační stránce. Proto bylo zapotřebí pokusit se nalézt řešení přibližné, které by bylo kompromisem mezi řešením optimálním a nároky na časovou odezvu u řešení. Matematické model heuristického řešení částečně vychází z matematického modelu řešení optimálního. Tyto dvě kapitoly daly vznik rozvrhovacímu algoritmu pro výslednou aplikaci. Vývojem aplikace samotné se zabývá kapitola čtvrtá (specifikace aplikace) a kapitola pátá (implementační část). V závěru páté kapitoly jsou uvedeny dosažené experimentální výsledky.



Obrázek 1.1: Metodika práce.

Kapitola 2

Nalezení optimálního řešení

Pro nalezení optimálního řešení byl vybrán jeden z mnoha nástrojů, který se zabývá řešením problémů lineárního programování. Byl zvolen nástroj GLPK [24], který je určen k řešení úloh matematického programování (lineárního programování (LP), smíšeného celočíselného programování (MIP) apod.). K tomuto projektu existuje poměrně dobrá dokumentace, a to jak programového rozhraní, tak implementovaných algoritmů a vnitřních datových struktur. V následujících části popíšu stručně ideu a syntaxi programování v tomto nástroji.

2.1 Syntaxe a programování

Programování spočívá ve vytvoření modelu, který bude co nejlépe odpovídat našemu systému, a to modelu matematického. Matematický model se vytvoří pomocí omezení, které pro náš systém musí platit.

2.1.1 Syntaxe

GLPK obsahuje jako každý jiný programovací jazyk klíčová slova (*and, or, if, then,..*), podporuje aritmetické operace, symbolická jména u proměnných atd. Začněme jednoduchým příkladem. Uvažujme optimalizační úlohu, kde cílem je rozmístit maximální počet královen na šachovnici o rozměrech $n \times n$. Zde je zdrojový kód:

```
/* QUEENS, a classic combinatorial optimization problem */
/* Written in GNU MathProg by Andrew Makhorin <mao@mai2.rcnet.ru> */
/* The Queens Problem is to place as many queens as possible on the 8x8
```

```
   (or more generally, nxn) chess board in a way that they do not fight
   each other. This problem is probably as old as the chess game itself,
   and thus its origin is not known, but it is known that Gauss studied
   this problem. */

param n, integer, > 0, default 30;
/* size of the chess board */

var x{1..n, 1..n}, binary;
/* x[i,j] = 1 means that a queen is placed in square [i,j] */

s.t. a{i in 1..n}: sum{j in 1..n} x[i,j] <= 1;
/* at most one queen can be placed in each row */

s.t. b{j in 1..n}: sum{i in 1..n} x[i,j] <= 1;
/* at most one queen can be placed in each column */

s.t. c{k in 2-n..n-2}: sum{i in 1..n, j in 1..n: i-j == k} x[i,j] <= 1;
/* at most one queen can be placed in each "\"-diagonal */

s.t. d{k in 3..n+n-1}: sum{i in 1..n, j in 1..n: i+j == k} x[i,j] <= 1;
/* at most one queen can be placed in each "/"-diagonal */

maximize obj: sum{i in 1..n, j in 1..n} x[i,j];
/* objective is to place as many queens as possible */

/* solve the problem */
solve;

/* and print its optimal solution */
for {i in 1..n}
{  for {j in 1..n} printf " %s", if x[i,j] then "Q" else ".";
   printf("\n");
}
 end;
```

V tabulce 2.1 jsou popsány jednotlivé příkazy z příkladu. Tyto příkazy byly postačující i při řešení NSP, proto jich zde není uvedeno více (kompletní dokumentaci naleznete v [24]). V úvodní části za komentářem následuje definice konstant a proměnných. Poté následuje množina omezení a definice cílové funkce a její optimalizace (minimalizace či maximalizace). Příkazem *solve* je spuštěn výpočet a na závěr vypsáno řešení pomocí příkazu *printf*. Zdrojový kód se spustí z příkazové řádky pomocí příkazu

```
>glpsol --model mymodel.mod
```

Tabulka 2.1: Příkazy v GLPK.

Název příkazu	Příkaz	Funkce příkazu
Comment	/* ... */	blokový komentář
Parameter statement	param	definice konstant
Variable statement	var	definice proměnných
Constraint statement	subject to s.t. subj to	definice omezení
Objective statement	minimize (maximize)	optimalizace cílové funkce
Solve statement	solve	spuštění výpočtu
Printf statement	printf	výpis řetězce na obrazovku
For statement	for	klasický cyklus for

Nástroj GLPK poskytuje mnoho možností. Jednou z nich je možnost ověření správnosti zdrojového kódu (nikoliv v oblasti syntaxe, ale v oblasti sémantiky). U problému s malou mohutností lze správnost matematického modelu ověřit tak, že model převedeme do popisu lineárního programování

```
>glpsol --wlpt mymodel.lp --model mymodel.mod
```

Jak taková struktura modelu vypadá, je popsáno v následující části.

2.1.2 Obecná definice problému LP v GLPK

Formulace problému LP ([3]) je následující:

Mimimalizuj (maximalizuj)

$$Z = c_1x_{m+1} + c_2x_{m+2} + ... + c_nx_{m+n} + c_0$$

s ohledem na lineární omezení

$$\begin{aligned} x_1 &= a_{11}x_{m+1} + a_{12}x_{m+2} + ... + a_{1n}x_{m+n} \\ x_2 &= a_{21}x_{m+1} + a_{22}x_{m+2} + ... + a_{2n}x_{m+n} \\ &\vdots \\ x_m &= a_{m1}x_{m+1} + a_{m2}x_{m+2} + ... + a_{mn}x_{m+n} \end{aligned}$$

a meze proměnných

$$\begin{array}{rcccl} l_1 & \leq & x_1 & \leq & u_1 \\ l_2 & \leq & x_2 & \leq & u_2 \\ & & \vdots & & \\ l_{m+n} & \leq & x_{m+n} & \leq & u_{m+n} \end{array} ,$$

kde symboly vyjadřují

Tabulka 2.2: Použité symboly u LP.

Symbol	Význam
Z	cílová funkce
$x_{m+1}, x_{m+2}, ..., x_{m+n}$	strukturální proměnné
$c_1, c_2, ..., c_n$	koeficienty v cílové funkci
c_0	konstantní člen v cílové funkci
$x_1, x_2, ..., x_m$	pomocné proměnné
$a_{11}, a_{12}, ..., a_{mn}$	koeficienty v lineárních omezeních
$l_1, l_2, ..., l_{m+n}$	spodní meze strukturálních proměnných
$u_1, u_2, ..., u_{m+n}$	horní meze strukturálních proměnných

2.2 Definice problému NSP

Řešením problému, který je znám pod pojmem „Nurse Scheduling Problem“ (NSP), je nalezení rozvrhu, který je v souladu se všemi omezeními, která tento problém vymezují.

2.2.1 Slovní formulace omezení

Omezení požadovaná nemocnicí

- den je rozdělen na dvě směny, denní a noční
- každá sestra pracuje 176 hodin za jeden rozvrh (4 týdny), což odpovídá 14,67 dne
- klasická pracovní doba se pohybuje mezi 14 a 15 dny za 4 týdny, přebytečné hodiny nad 176 pracovních hodin jsou počítány jako přesčasy
- žádná sestra nesmí pracovat více jak čtyři dny po sobě

- žádná sestra by neměla v žádném případě pracovat dvě směny během jednoho dne (zahrnuje i kombinaci noční směna-ranní směna následující den)
- na každé směně musí být alespoň minimální stanovený počet sester)
- noční směny musí být zastoupeny u každé ze sester alespoň 25 %

Omezení založené na ergonomii

- nepřípustné izolované pracovní dny (volno-směna-volno)
- nepřípustné izolované volné dny (směna-volno-směna)
- nepřípustná kombinace směn (denní směna-noční směna následující den)
- všechny sestry by měly mít přibližně stejný počet nočních směn a volných víkendů
- počet denních směn by měl převyšovat počet nočních směn u každé sestry

2.2.2 Matematický model

Na základě výše zmíněných omezení byl vytvořen matematický model [25], který se skládá ze dvou typů omezení; nutných omezení (*hard constraints*) a doplňujících omezení (*soft constraints*). První typ je soubor podmínek, které musí být splněny v každém případě. Druhým typem jsou podmínky podmiňovacího charakteru, tzn. pokud nebudou splněny, dojde k penalizaci v cílové funkci Z. Optimalizace NSP spočívá v minimalizaci penalizace, tzn. splnění všech nutných omezení a co největšího množství doplňujících omezení s ohledem na váhově navrženou penalizaci v cílové funkci (různým podmínkám je přiřazena různá váha penalizace v Z). Tyto váhy jsou přiděleny na základě zkušeností z nemocnic.

2.2.2.1 Použité proměnné v matematickém modelu

Tabulka 2.3: Použité proměnné v matematickém modelu pro GLPK.

Proměnná	Význam
n	počet dní v rozvrhu
m	počet sester v nemocnici
i	index pro dny, $i = 1, 2, ..., n$
k	index pro sestry, $k = 1, 2, ..., m$
D_i	požadovaný počet sester pro denní směnu dne i
N_i	požadovaný počet sester pro noční směnu dne i
$XD_{i,k} = 1$	sestra k slouží denní směnu dne $i, \forall i, \forall k$
$XD_{i,k} = 0$	sestra k neslouží denní směnu dne $i, \forall i, \forall k$
$XN_{i,k} = 1$	sestra k slouží noční směnu dne $i, \forall i, \forall k$
$XN_{i,k} = 0$	sestra k neslouží noční směnu dne $i, \forall i, \forall k$
$XR_{i,k} = 1$	sestra k má volno dne $i, \forall i, \forall k$
$XR_{i,k} = 0$	sestra k nemá volno (slouží) dne $i, \forall i, \forall k$

2.2.2.2 Formulace nutných omezení

Nutná omezení jsou taková, která musí být vždy dodržena. Daná omezení jsou napsána pro rozvrh trvající 28 dní. V případě jiného počtu se musí podmínky (2.6), (2.7), (2.8), (2.9) upravit, konkrétně indexy a pravé strany rovnic. Podmínky (2.1), (2.2) popisují zabezpečení minimálního počtu sester na denní a noční směně.

$$\sum_{k=1}^{m} XD_{i,k} \geq D_i, \forall i \tag{2.1}$$

$$\sum_{k=1}^{m} XN_{i,k} \geq N_i, \forall i \tag{2.2}$$

Omezení (2.3), (2.4) zakazují pracovat dvě po sobě jdoucí směny po sobě.

$$XD_{i,k} + XN_{i,k} + XR_{i,k} = 1, \forall i, \forall k \tag{2.3}$$

$$XD_{i+1,k} + XN_{i,k} \leq 1, \forall i = 1, n-1, \forall k \tag{2.4}$$

Omezení (2.5) zamezí tomu, že každá sestra nebude pracovat více jak čtyři dny po sobě, tj. bude mít alespoň jeden den volna v okně přes pět dní.

$$XR_{i,k} + XR_{i+1,k} + XR_{i+2,k} + XR_{i+3,k} + XR_{i+4,k} \geq 1, \forall i = 1, n-4, \forall k \tag{2.5}$$

Další podmínka (2.6) zajistí během víkendů v rozvrhu (28 dní $\sim$ 4 víkendy) alespoň 4 dny volna.

$$XR_{6,k} + XR_{7,k} + XR_{13,k} + XR_{14,k} + XR_{20,k} + XR_{21,k} + XR_{27,k} + XR_{28,k} \geq 4, \forall k \quad (2.6)$$

Dvojice podmínek (2.7) a (2.8) vymezují počet odpracovaných směn za rozvrh (minimálně 14, maximálně 16 odpracovaných směn).

$$\sum_{i=1}^{n} (XD_{i,k} + XN_{i,k}) \geq 14, \forall k \quad (2.7)$$

$$\sum_{i=1}^{n} (XD_{i,k} + XN_{i,k}) \leq 16, \forall k \quad (2.8)$$

Poslední podmínka (2.9) definuje poměr denních a nočních směn, v našem případě je uvažováno minimálně 25 % nočních směn (4 směny z celkového počtu 16-ti směn).

$$\sum_{i=1}^{n} XN_{i,k} \geq 4, \forall k \quad (2.9)$$

2.2.2.3 Formulace doplňujících omezení

Doplňující omezení jsou taková, na kterých se optimalizuje NSP a hledá se nejlepší rozvrh, který bude mít nejmenší penalizaci. V tabulce 2.4 jsou zavedeny další pomocné proměnné, právě pro penalizaci.

Tabulka 2.4: Použité proměnné v matematickém modelu pro GLPK (2).

Proměnná	Význam
$d1_k^+ > 0$	pozitivní odchylka od počtu pracovních dní v rozvrhu $\Rightarrow d1_k^- = 0$
$d1_k^- > 0$	negativní odchylka od počtu pracovních dní v rozvrhu $\Rightarrow d1_k^+ = 0$
$d2_k^+ > 0$	pozitivní přesah denní směny-noční směny $\Rightarrow d2_k^- = 0$
$d2_k^- > 0$	negativní přesah denní směny-noční směny $\Rightarrow d2_k^+ = 0$
$d3_k^+ = 1$	více jak jedna ze směn denní-noční následující den $\Rightarrow d3_k^- = 0$
$d3_k^- = 1$	pouze jedna ze směn denní-noční následující den $\Rightarrow d3_k^+ = 0$

Proměnná	Význam
$d4_k^+ = 1$	nesplněna podmínka na izolované dny volna $\Rightarrow d4_k^- = 0$
$d4_k^- = 1$	splněna podmínka na izolované dny volna $\Rightarrow d4_k^+ = 0$
$d5_k^+ = 1$	nesplněna podmínka na izolované pracovní dny $\Rightarrow d5_k^- = 0$
$d5_k^- = 1$	splněna podmínka na izolované pracovní dny $\Rightarrow d5_k^+ = 0$

Podmínka (2.10) minimalizuje odchylku mezi sumou odpracovaných dnů a požadovaných dnů. Také zde platí, že je tato podmínka formulována pro 28-denní rozvrh ($\sim$ 15 pracovních dní). Penalizována je kladná odchylka $d1_k^+$.

$$\sum_{i=1}^{n}(XD_{i,k} + XN_{i,k}) - (d1_k^+ - d1_k^-) = 15, \forall k \tag{2.10}$$

Druhé omezení (2.11) spočívá v zajištění většího počtu denních směn než nočních. Zde je penalizována naopak kladná odchylka $d2_k^-$.

$$\sum_{i=1}^{n} XD_{i,k} - \sum_{i=1}^{n} XN_{i,k} - (d2_k^+ - d2_k^-) = 1, \forall k \tag{2.11}$$

Další omezení (2.12) zamezuje tomu, aby sestra pracovala denní směnu a noční směnu následující den. Zde je opět penalizována kladná odchylka $d3_k^+$.

$$XD_{i,k} + XN_{i+1,k} - (d3_k^+ - d3_k^-) = 1, \forall i, i = 1, n-1, \forall k \tag{2.12}$$

Poslední dvě omezení se zabývají izolovanými dny. První z nich (2.13) minimalizuje počet izolovaných dní volna. Druhé (2.14) naopak minimalizuje počet izolovaných pracovních dní. Penalizovány jsou kladné odchylky $d4_k^+$ a $d5_k^+$.

$$XR_{i,k} + XD_{i+1,k} + XN_{i+1,k} + XR_{i+2,k} - (d4_k^+ - d4_k^-) = 1, \forall i, i = 1, n-2, \forall k \tag{2.13}$$

$$XD_{i,k} + XN_{i,k} + XR_{i+1,k} + XD_{i+2,k} + XN_{i+2,k} - (d5_k^+ - d5_k^-) = 1, \forall i, i = 1, n-2, \forall k \tag{2.14}$$

2.2.2.4 Stanovení cílové funkce

Cílovou funkcí nazýváme takovou funkci, kterou chceme optimalizovat. Naše cílová funkce se bude sestávat ze součtu penalizací, které budou váhově odlišeny, protože klademe různý důraz na jednotlivá omezení. V tabulce 2.5 jsou uvedeny jednotlivé váhy, které byly použity při řešení.

Tabulka 2.5: Váhy pro penalizaci v matematickém modelu.

váha	p_1	p_2	p_3	p_4	p_5
hodnota	20	5	3	1	1

Výsledná cílová funkce je

$$Z = p_1 \sum_{k=1}^{m} d1_k^+ + p_2 \sum_{k=1}^{m} d2_k^- + p_3 \sum_{i=1}^{n-1} \sum_{k=1}^{m} d3_k^+ + p_4 \sum_{i=1}^{n-2} \sum_{k=1}^{m} d4_k^+ + p_5 \sum_{i=1}^{n-2} \sum_{k=1}^{m} d5_k^+ \quad (2.15)$$

obecně tedy

$$Z = p_1 Z_1 + p_2 Z_2 + p_3 Z_3 + p_4 Z_4 + p_5 Z_5 \quad , \quad (2.16)$$

zkráceně

$$Z = \sum_j p_j \, Z_j \quad , \quad (2.17)$$

kde j označuje j-tou složku Z, $j = 1, 2, ..., \dim(Z)$, v tomto případě $j = 1, 2, ..., 5$.

Usilujeme o

$$\min(Z) \quad , \quad (2.18)$$

v ideálním případě, kdy jsou všechny (nutné i doplňující) podmínky splněny

$$Z = 0 \quad . \quad (2.19)$$

2.3 Výsledky

Popsaný matematický model (zdrojové kódy - viz příloha A) byl otestován v několika variacích. Volné proměnné, které se mohou měnit, jsou n, m, D_i, N_i. Pro jednoduchost bylo uvažováno, že $D_i = N_i$, ale pokud by byl minimální počet sester na denní a noční směně různý, nic by to neměnilo na složitosti problému. Další možnost, která se nabízí, je brát v potaz pouze některé doplňující omezení a některé vynechat a nezahrnovat do cílové funkce Z. Váhy v cílové funkci měněny nebyly (2.5). Nejprve jsou uvedeny výsledky odpovídající modelu se všemi doplňujícími omezeními.

Tabulka 2.6: Výsledky z GLPK.

Doplňující podmínky	n	m	$D_i = N_i$	Z	t_s [s]
všechny	28	8	2	0	1884,2
(2.10), (2.11), (2.12)	28	8	2	0	11,1

Z tabulky vidíme, že pokud snížíme počet doplňujících omezení o dvě, navíc o dvě nejméně vážená, která se zabývají izolovanými dny, dojde k podstatnému zrychlení výpočtu. V další části jsou již doplňující omezení zredukovány na tři ((2.10), (2.11), (2.12)). Podoba cílové funkce Z se tedy změní na

$$Z = p_1 \sum_{k=1}^{m} d1_k^+ + p_2 \sum_{k=1}^{m} d2_k^- + p_3 \sum_{i=1}^{n-1} \sum_{k=1}^{m} d3_k^+ \quad . \tag{2.20}$$

2.3.1 Výsledky pro matematický model se třemi doplňujícími omezeními

V tabulce 2.7 jsou uvedeny hodnoty dob rozvrhování t_s a cílové funkce Z pro rozvrhy na 28 a 14 dní, které byly vytvořeny pomocí matematického modelu, který má pouze doplňující omezení (2.10), (2.11), (2.12).

Tabulka 2.7: Výsledky z GLPK (2).

Doplňující podmínky		**Počet dní**			
		28		14	
(2.10), (2.11), (2.12)		Z	t_s [s]	Z	t_s [s]
...	m	$D_i = N_i = 2$			
...	6	∞	-[1]	∞	-[1]
...	7	175	380,2	35	3,3
...	8	0	11,1	20 (0)	> 1h[2]
...	10	0	27,2	0	126,0

[1] Rozvrh nelze navrhnout.

[2] Výpočet byl po jedné hodině přerušen, hodnota Z je hodnota po jedné hodině výpočtu (v závorce je uvedena hodnota min(Z), ke které by výpočet při dostatku času dospěl).

Doplňující podmínky (2.10), (2.11), (2.12)		**Počet dní**			
		28		14	
	m	Z	t_s [s]	Z	t_s [s]
...	12	0	33,7	0	2,9
...	14	0	29,0	0	4,9
...	16	0	34,0	0	6,8
...	18	0	42,3	0	13,5
...	20	0	46,2	0	13,5
...	22	0	61,5	0	18,3
...	28	0	99,3	0	41,0
...	m	$D_i = N_i = 3$			
...	8	∞	-[1]	∞	-[1]
...	10	∞	-[1]	130	5,0
...	11	115	175,3	45 (35)	> 1h[2]
...	12	0	42,6	30 (0)	> 1h[2]
...	14	0	70,0	15 (0)	> 1h[2]
...	16	0	120,7	0	9,4
...	18	0	109,2	0	15,9
...	20	0	70,9	0	17,3
...	22	0	85,0	0	44,6
...	28	0	172.8	0	46,7
...	m	$D_i = N_i = 4$			
...	12	∞	-[1]	∞	-[1]
...	13	∞	-[1]	225	8,7
...	14	358 (350)	> 1h[1]	70	39,0
...	16	0	95,3	40 (0)	> 1h[1]
...	18	0	142,7	25 (0)	> 1h[1]
...	20	0	155,7	5 (0)	> 1h[1]
...	22	0	278,2	6 (0)	> 1h[1]
...	28	0	326,4	0	32,6

Dále zde uvedu pro ilustraci výpis při výpočtu v GLPK, konkrétně se jedná o rozvrh na 28 dní, k dispozici je 12 sester a na každé směně je vyžadována přítomnost 3 sester.

```
>glpsol --model pok28-12-3.mod
Reading model section from pok28-12-3.mod...
158 lines were read...
Generating hc1...
Generating hc2...
Generating hc3...
Generating hc4...
Generating hc5...
Generating hc7...
Generating hc8...
Generating hc9...
Generating sc1...
Generating sc2...
Generating sc3...
Generating obj...
Model has been successfully generated lpx_simplex: original LP has 1389 rows,
1704 columns, 8484 non-zeros lpx_simplex: presolved LP has 1388 rows,
1356 columns, 7788 non-zeros lpx_adv_basis: size of triangular part = 1388
    200:   objval = -2.249000000e+003   infeas =  4.947447447e-001 (0)
    400:   objval = -4.462272727e+002   infeas =  1.850515288e-001 (0)
    600:   objval = -1.283333333e+001   infeas =  7.845345345e-002 (0)
    770:   objval =  2.195000000e+002   infeas =  3.128753513e-018 (0)
*   770:   objval =  2.195000000e+002   infeas =  2.778333119e-015 (0)
*   800:   objval =  1.733308698e+002   infeas =  4.682302528e-015 (0)
*  1000:   objval =  6.754800442e-001   infeas =  1.492088883e-034 (0)
*  1004:   objval =  2.997602166e-014   infeas =  0.000000000e+000 (0)
OPTIMAL SOLUTION FOUND
Integer optimization begins...
Objective function is integral
+  1004: mip =     not found yet >=              -inf        (1; 0)
+  1244: mip =     not found yet >=  0.000000000e+000        (11; 0)
+  1438: mip =     not found yet >=  0.000000000e+000        (19; 0)
+  1649: mip =     not found yet >=  0.000000000e+000        (28; 0)
+  1849: mip =     not found yet >=  0.000000000e+000        (37; 0)
+  2175: mip =     not found yet >=  0.000000000e+000        (46; 0)
+  2612: mip =     not found yet >=  0.000000000e+000        (56; 0)
+  3071: mip =     not found yet >=  0.000000000e+000        (69; 0)
+  4021: mip =     not found yet >=  0.000000000e+000        (91; 0)
+  4042: mip =  0.000000000e+000 >=  0.000000000e+000   0.0\% (91; 0)
+  4042: mip =  0.000000000e+000 >=     tree is empty   0.0\% (0; 181)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   42.6 secs
Memory used: 3.3M (3451011 bytes)

Number of days in schedule: 28
Number of nurses: 12
Required staff for each shift: 3
```

```
   day 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8

 nurse1 N N N N - D D - D - - D D D D - D - - N - - - - N - N -  15|(8+7)
 nurse2 N - - D - - D D D D - D D - D - N - N N N - - - - - N N  15|(8+7)
 nurse3 - N - D D D D - N - D - - N - D D - D - N N - - N N - -  15|(8+7)
 nurse4 - N - D - D - D - N - D D D - D D - - - - - N N N N - N  15|(8+7)
 nurse5 N - D D D - N - D - - N - - N - - D D D D - N N - - - N  15|(8+7)
 nurse6 D - N - N N - N N - D - - - - N - D D D D - D D - N - -  15|(8+7)
 nurse7 - D D - N N N N - D D D - D D - - N - D - N - - - - N -  15|(8+7)
 nurse8 - - N N - D - N N N N - N - - D - D - - - D D D - D - D  15|(8+7)
 nurse9 D D - N N - - - - D - - N N N N - N - - - D - D D D D -  15|(8+7)
nurse10 D D - D - N N - D - N N N - - N N - - - D D - - D - D -  15|(8+7)
nurse11 D D D - D - - D - N N N - - N - N - N - N - D - D - - D  15|(8+7)
nurse12 D D - - D D - - - D - - - N - - - N N N - N N N - D D D  15|(8+7)

dayS    5 5 3 5 4 5 3 3 4 4 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
nightS  3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Model has been successfully processed
```

2.4 Zhodnocení

K rozvržení služeb byl využit nástroj GLPK, řešící tento problém celočíselným lineárním programovaním. Stavový prostor řešení procházel pomocí algoritmu Branch&Bound, z čehož lze vyvodit několik důsledků.

- hledání řešení problému není efektivní z hlediska času
- čas potřebný ke hledaní řešení není lineárně závislý na „obtížnosti“ (mohutnosti, složitosti) problému; je to způsobeno tím, že při používání algoritmu Branch&Bound záleží na prořezávání větví stromu řešení (tzn. i „složitější“ problém může být vyřešen dříve, pokud se nám podaří dostatečně rychle zredukovat stavový prostor řešení)

Z testů (2.7) si můžeme povšimnout, že kvalitnějších výsledků bylo dosaženo pro 28-denní rozvrh. U 14-ti denního rozvrhu jsou časy značně kolísavé, a to patrně z toho důvodu, že počet dní je malý – nelze však jednoznačně říci, že by byla doba rozvrhování t_s pro 14 dní kratší než pro 28 dní. Doba rozvrhování t_s je závislá nejenom na n (počet dní), ale také na vytíženosti sester, tj. poměrech

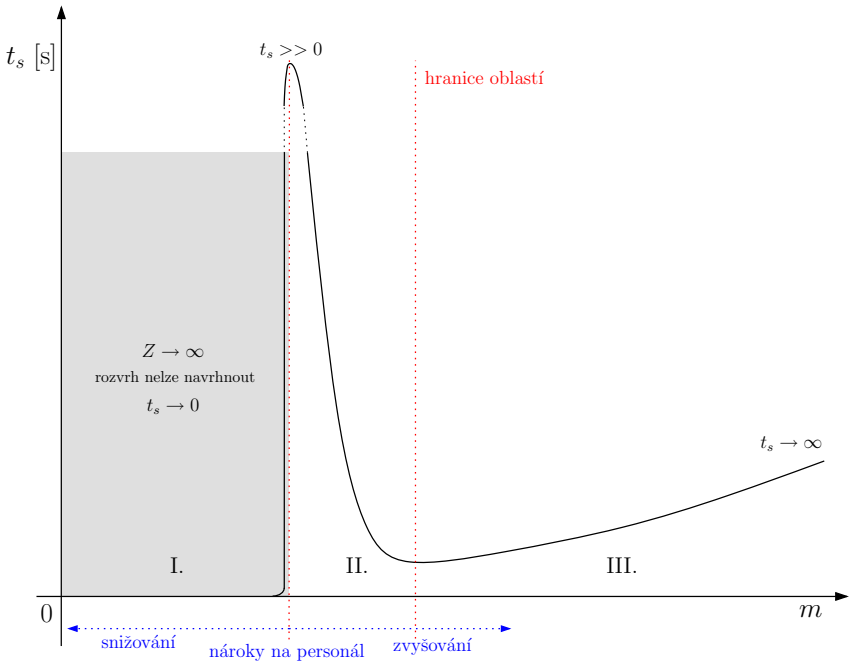
$$\frac{D_i}{m}, \frac{N_i}{m} \tag{2.21}$$

Čím jsou tyto poměry vyšší, tím obtížněji je navrhnutelný rozvrh. Teoretická velikost doby rozvrhování je zobrazena na obr. 2.1. Graf by se dal rozdělit do tří částí:

I. Tato oblast zahrnuje případy, kdy volné parametry (n, m, D_i, N_i, nutná omezení, doplňující omezení) jsou zvoleny tak, že rozvrh není možné sestavit. Čas potřebný ke zjištění této skutečnosti se blíží k nule. Hranice oblastí I. a II. (červeně) se pohybuje při konstantním m se změnou parametrů D_i, N_i (modře). Kolem této hranice je doba rozvrhování pro malé problémy konečná, pro velké problémy se blíží k nekonečnu, a to z obou stran.

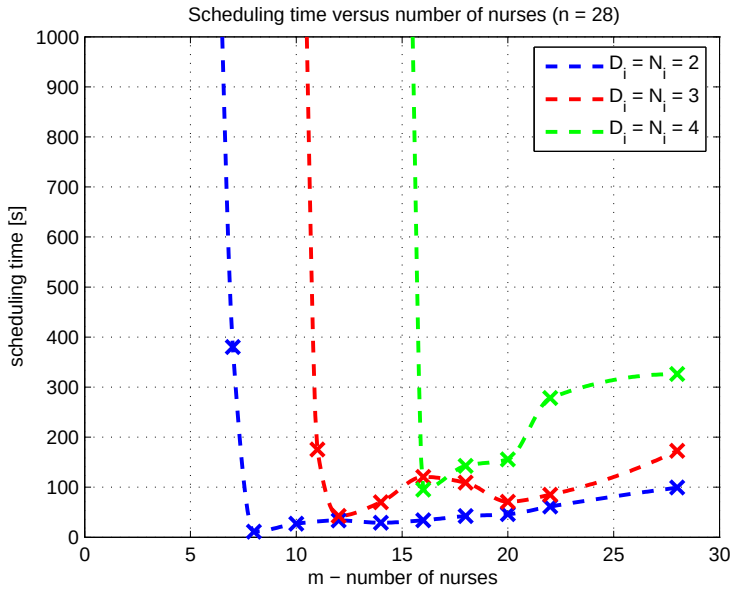
II. Prostřední část grafu odpovídá oblasti, kdy již rozvrh zhotovit lze, ale jeho nalezení je poměrně složité. Tato oblast je vůči další oblasti poměrně úzká a křivka zde velmi rychle klesá. Na hranici oblastí II. a III. (červeně) se nachází rozvrh s minimálním rozvrhovacím časem a také podobně zde dochází při konstantním m k posunu hranice se změnou D_i, N_i (modře).

III. V této části doba rozvrhování exponenciálně roste s počtem sester (m).



Obrázek 2.1: Teoretická závislost výpočetního času rozvrhu na počtu sester.

Pro srovnání jsem zde uvedl i závislost (obr. 2.2) získanou z testů, a to konkrétně pro rozvrhy na 28 dní. Pro přehledost zde nejsou vyneseny hodnoty, které patří do oblasti I., jejich hodnoty se blíží nule.



Obrázek 2.2: Závislost výpočetního času rozvrhu na počtu sester získaná z testů.

Všechny testy byly spušteny na AMD Athlon XP 1800+@512MB DDR RAM v prostředí GLPK 4.8. Protože časy pro sestavení rozvrhu nejsou zrovna uspokojivé, je třeba zvolit kompromis mezi „kvalitou“ řešení a časem potřebným pro získání řešení. Další kapitola je proto věnována algoritmu, který sice nenalezne optimální řešení NSP, ale vyhoví běžným požadavkům.

Kapitola 3

Nalezení heuristického řešení

3.1 Úvod

Cílem této kapitoly bylo nalézt takové řešení NSP, aby řešilo nevýhody optimálního řešení uvedené v kapitole 2. Protože toto řešení nebude optimální, nazvěme jej heuristickým (přibližným). Hlavním požadavkem a výhodou heuristického řešení oproti řešení optimálnímu je kratší výpočetní čas, nevýhodou naopak, že nalezený rozvrh není zcela optimální. Tato kapitola popisuje navržené řešení a diskutuje jeho vlastnosti.

3.2 Matematický model heuristického řešení

Matematický model heuristického řešení vychází z matematickému modelu řešení optimálního, jsou zde ale dva zásadní rozdíly. U optimálního řešení byl matematický model sestrojen pro dvousměnný provoz (12-ti hodinové denní a noční směny XD, XN). U heuristického řešení je uvažován provoz třísměnný (ranní směny XE, odpolední směny XL a směny noční XN). Názvy proměnných byly voleny s ohledem na zavedené názvy z části 2.2.2. Druhým rozdílem je volba penalizačních vah u doplňujících omezení. Hodnoty vah nevypovídají na první pohled o tom, které kritérium je více bráno v úvahu. To je způsobeno tím, že výpočet hodnoty Z by měl být nezávislý na velikosti (složitosti) problému, tj. nezávislý na parametrech n, m. Proto jsou proměnné Z_j vystupující jako suma matice či vektoru pro penalizaci děleny (normovány) dimenzí matice či vektoru (dim_j). Po vydělení vznikne skalár, který je násoben již penalizační vahou p_j. Funkčnost algoritmu je poměrně citlivá na nastavení vah p_j.

3.2.1 Formulace nutných omezení

Nejdůležitější nutnou podmínkou je podmínka týkající se zabezpečení dostatečného počtu sester na směně. Adekvátně k (2.1), (2.2) zde vystupují (3.1), (3.2), (3.3).

$$\sum_{k=1}^{m} XE_{i,k} \geq RE, \forall i \tag{3.1}$$

$$\sum_{k=1}^{m} XL_{i,k} \geq RL, \forall i \tag{3.2}$$

$$\sum_{k=1}^{m} XN_{i,k} \geq RN, \forall i \tag{3.3}$$

Tato omezení jsou důležitá z hlediska provozu nemocnice, další dvě následující omezení (3.4), (3.5) (alternativa k (2.3), (2.4)) jsou důležitá spíše pro sestry. Jedná se o zamezení více jak jedné směny denně či souvislého bloku noční směny a ranní směny následujícího dne.

$$XE_{i,k} + XL_{i,k} + XN_{i,k} + XR_{i,k} = 1, \forall i, \forall k \tag{3.4}$$

$$XE_{i+1,k} + XN_{i,k} \leq 1, \forall i = 1, n-1, \forall k \tag{3.5}$$

Nutná podmínka (2.5) byla v tomto modelu převedena na podmínku doplňující (3.7) s velmi vysokou penalizací. Převod byl proveden na základě zkušeností získaných během implementace (snaha zachovat toto omezení mezi nutnými vedla na obtížné a těžkopádné provádění inicializace i optimalizace).
Pro omezení na maximálně čtyři pracovní dny za sebou musí platit:

$$-XR_{i,k} - XR_{i+1,k} - XR_{i+2,k} - XR_{i+3,k} - XR_{i+4,k} + 1 \leq 0, \forall i = 1, n-4, \forall k \tag{3.6}$$

$$-\sum XR_{i..\,i+4,\,k} + 1 - (d0_k^+ - d0_k^-) = 0, \forall i = 1, n-4, \forall k \tag{3.7}$$

Penalizuje se pouze v případě, kdy $d0_k^+ = 1$ (tab. 3.1).

Tabulka 3.1: Penalizace matematickém modelu pro heuristického řešení.

Symbol	Význam
$d0_k^+ = 1$	pozitivní odchylka v kritériu, penalizována $\Rightarrow d0_k^- = 0$
$d0_k^- \geq 0$	negativní odchylka v kritériu, bez penalizace $\Rightarrow d0_k^+ = 0$

3.2.2 Formulace doplňujících omezení

Počet doplňujících omezení se nám oproti minulé kapitole nezmenšil, naopak zvětšil. Kromě omezení (3.7) zde máme také alternativní omezení k (2.10), (2.11), (2.12), (2.13), (2.14). V odpovídajícím pořadí jsou uvedeny níže.

Podmínka (3.8) penalizuje přesčasy sester a zabezpečuje poměr mezi volnem a pracovními dny (počet pracovních dnů by měl tvořit zhruba polovinu z n dnů).

$$\sum_{i=1}^{n} XE_{i,k} + \sum_{i=1}^{n} XL_{i,k} + \sum_{i=1}^{n} XN_{i,k} - (d1_k^+ - d1_k^-) = round\left(\frac{n}{2}\right), \forall k \tag{3.8}$$

Omezení (3.9) slouží k vyvážení jednotlivých typů směn.

$$\sum_{i=1}^{n} XE_{i,k} + \sum_{i=1}^{n} XL_{i,k} - \sum_{i=1}^{n} XN_{i,k} - (d2_k^+ - d2_k^-) = round\left(\frac{3n}{10}\right), \forall k \tag{3.9}$$

Omezení (3.10) zamezuje následnost ranní směny a noční směny následujícího dne.

$$XE_{i,k} + XN_{i+1,k} - (d3_k^+ - d3_k^-) = 1, \forall i = 1, n-1, \forall k \tag{3.10}$$

Poslední dvě doplňující podmínky ((3.11), (3.12)) se soustředí na redukci izolovaných dnů, a to za podmínky 3.13.

$$XR_{i,k} + XE_{i+1,k} + XL_{i+1,k} + XN_{i+1,k} + XR_{i+2,k} - (d4_k^+ - d4_k^-) = 2 \tag{3.11}$$

$$XE_{i,k} + XL_{i,k} + XN_{i,k} + XR_{i+1,k} + XE_{i+2,k} + XL_{i+2,k} + XN_{i+2,k} - (d5_k^+ - d5_k^-) = 2 \tag{3.12}$$

$$\forall i = 1, n-2, \forall k \tag{3.13}$$

Penalizace se provádí stejně jako u optimálního řešení (tab. 2.4).

3.2.3 Stanovení cílové funkce Z

Podobně jako v předchozí kapitole je Z součet jednotlivých vážených penalizací. Je zde však rozdíl ve vážení těchto příspěvků. Příspěvky jsou úmyslně váženy (normovány) také dimenzemi jednotlivých příspěvků, čímž se zamezí závislosti kritéria na velikosti problému.

Váhy p_j byly experimentálně nastaveny takto

Tabulka 3.2: Váhy pro penalizaci v matematickém modelu.

index j	0	1	2	3	4	5
p_j	200	5	3	10	20	20
dim_j	$m(n-4)$	m	m	$m(n-1)$	$m(n-2)$	$m(n-2)$

Cílová funkce Z je

$$Z = \frac{p_0}{dim_0}\sum_{i=1}^{n-4}\sum_{k=1}^{m} d0_k^+ + \frac{p_1}{dim_1}\sum_{k=1}^{m} d1_k^+ + \frac{p_2}{dim_2}\sum_{k=1}^{m} d2_k^- +$$

$$+\frac{p_3}{dim_3}\sum_{i=1}^{n-1}\sum_{k=1}^{m} d3_k^+ + \frac{p_4}{dim_4}\sum_{i=1}^{n-2}\sum_{k=1}^{m} d4_k^+ + \frac{p_5}{dim_5}\sum_{i=1}^{n-2}\sum_{k=1}^{m} d5_k^+ \quad , \tag{3.14}$$

tedy

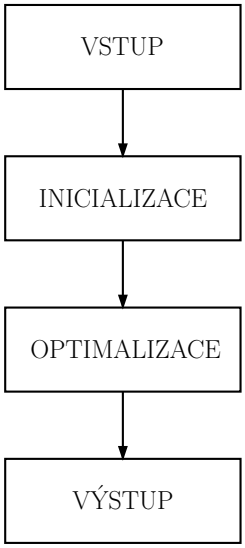
$$Z = \sum_j \frac{p_j}{dim_j} Z_j \quad . \tag{3.15}$$

Cílem je minimalizace Z

$$\min(Z) \quad . \tag{3.16}$$

3.3 Návrh algoritmu

Návrh algoritmu částečně vycházel z matematického modelu z [25] a [17]. V této části jsou popsány hlavní myšlenky, jak byl algoritmus navrhován (v části 3.4 uveden pseudokód a podrobnější blokové schéma závislosti jednotlivých funkcí obr. 3.5). Algoritmus se skládá ze čtyř částí (obr. 3.1).



Obrázek 3.1: Blokové schéma algoritmu.

3.3.1 Vstupní blok algoritmu

První blok (VSTUP) pokrývá zadání a kontrolu (celočíselnost, kladnost) vlastních vstupních parametrů pro tvorbu rozvrhu (tab. 3.3).

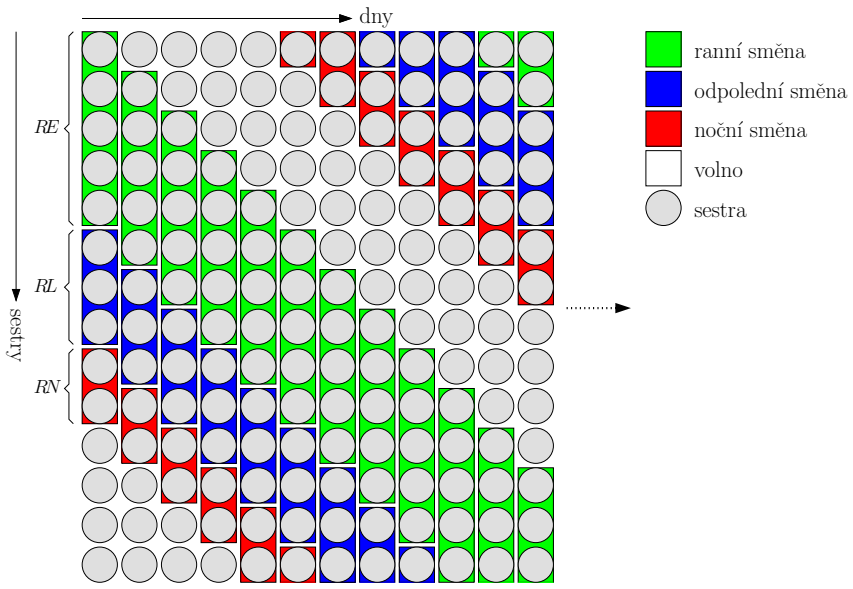
Tabulka 3.3: Vstupní parametry algoritmu heuristického řešení.

Symbol	Význam
n	počet dní v rozvrhu
m	počet sester v nemocnici
RE	minimální počet sester na ranní směně
RL	minimální počet sester na odpoledni směně
RN	minimální počet sester na noční směně
$weekday$	den v týdnu, kterým začíná rozvrh

3.3.2 Inicializační blok algoritmu

V bloku INICIALIZACE je třeba vygenerovat rozvrh, který bude již přípustný, tzn. bude splňovat nutná omezení (*hard constraints*). Při návrhu tohoto bloku bylo nutné vzít v úvahu, že tvorba počátečního rozvrhu hraje zásadní roli pro další optimalizační blok. Při rozmístění jednotlivých směn sestrám bylo třeba uvažovat rovnoměrnost rozmístění,

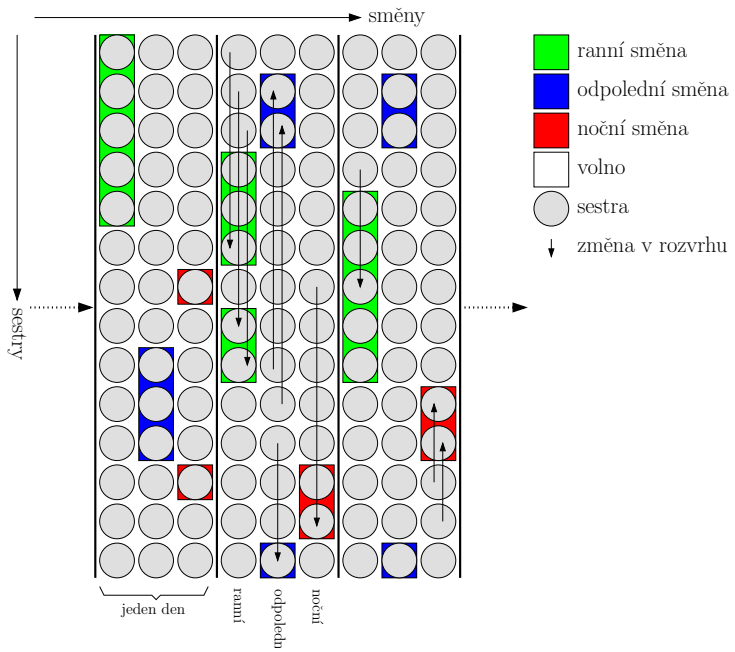
pokrýt minimální požadavky na počet sester na směně a dále předejít situaci, kdy jedna sestra bude pracovat více jak jednu směnu denně či pracovat noční směnu v návaznosti na ranní směnu následujícího dne. Nejvíce se nakonec osvědčil způsob cyklického přidělování směn sestrám, jak ilustruje obr. 3.2. Směny jsou přidělovány každý den tak, že nejdříve se přiřadí sestrám minimální počet ranních směn (*RE*), dále se přiřadí minimální počet odpoledních směn (*RL*) a nakonec se přiřadí směny noční (*RN*). Při obsazování dalšího dne se směny začínají obsazovat od následující sestry (vystupuje v rozvrhu pod indexem o jedna větším, než sestra, u které se začínalo s inicializací předchozího dne). Součástí toho bloku je také výpočet počáteční hodnoty cílové funkce Z.



Obrázek 3.2: Tvorba počátečního přípustného rozvrhu.

3.3.3 Optimalizační blok algoritmu

Blok OPTIMALIZACE zahrnuje minimalizaci cílové funkce Z. K optimalizaci dochází prohazováním směn mezi jednotlivými sestrami. Princip optimalizace spočívá v tom, že jsou prováděny pouze změny ve vertikálním směru (v rámci jedné směny – obr. 3.3) – tím nedojde k porušení jednoho z hlavních nutných omezení, a to minimálního počtu sester na směně.



Obrázek 3.3: Optimalizace na základě vertikálních změn směn mezi sestrami.

Optimalizace probíhá na podmínce s největším příspěvkem do Z. Najde se takový kandidát na změnu (prohození směny mezi dvěma sestrami v rámci jedné pracovní směny), u kterého by po změně nastalo největší zlepšení cílové funkce Z. Při optimalizaci je využita *metaheuristika Tabu Search* [26].

3.3.3.1 Metaheuristika Tabu Search

Pro nalezení přípustného řešení s vlastností, že hodnota cílové funkce Z nemusí být optimální, se využívají heuristické metody. Heuristické metody jsou narozdíl od exaktních metod (nacházejících optimální řešení) rychlé a nevykazují obtíže i při řešení rozsáhlejších problémů. Jsou formulovány pro konkrétní úlohu, existují však obecné heuristické postupy, které je možné aplikovat na obecný optimalizační problém. Takové heuristické metody nazýváme metaheuristiky. Nejznámější metaheuristiky jsou metoda lokálního hledání, metoda Tabu Search, metoda prahové akceptace či genetické algoritmy. Obecný princip metaheuristik je následující:

Mějme obecný optimalizační problém na množině přípustných řešení X. Hledáme optimální řešení $x^* \in X$, které minimalizuje cílovou funkci Z na množině X.

Předpokládejme, že pro každé $x \in X$ je definována množina $N(x) \subset X$, kterou nazývejme okolím bodu x, pro níž platí $x \in N(x)$.

Metoda označovaná jako Tabu Search řeší optimalizační problémy zavedením seznamu řešení, která již byla při postupném prohledávání množiny přípustných řešení zpracována. Tento seznam je označován jako tabu seznam (*Tabu List*) a obsahuje tedy řešení, ke kterým se nemáme již vracet, protože již byla zpracována. Leží již na posloupnosti řešení tvořené jednotlivými kroky metody. Při postupném hledání řešení je brán v potaz seznam zakázaných řešení *Tabu List* a přesouváme se pouze do takového řešení v okolí posledního řešení, které není v *Tabu List*. Tato metaheuristika narozdíl od metody lokálního hledání zamezuje uvíznutí v lokálním minimu při optimalizaci, a to zmíněným seznamem *Tabu List*.

Algoritmus 1 Metaheuristika Tabu Search

Krok 1

Zvolme výchozí připustné řešení x a položme $x^* := x$ (x^* označuje nejlepší zatím dosažené řešení).

Krok 2

Nalezneme $x' \in \overline{N}(x) \subset N(x)$, které nabývá minima funkce Z na množině $\overline{N}(x)$. Množina $\overline{N}(x)$ obsahuje ta řešení z $N(x)$, která nejsou v seznamu *Tabu List*.

Krok 3

Zapišme řešení x do seznamu *Tabu List* a dosaďme $x := x'$. Pokud je $Z(x) < Z(x^*)$, pak dosaďme $x^* := x$.

Krok 4

Pokud je splněno ukončovácí kritérium, pak výpočet končí (výsledkem je řešení x^*), jinak krok 2.

Protože se seznam tabuizovaných řešení postupně zvětšuje, v praxi se velikost seznamu *Tabu List* omezuje na určitou velikost. *Tabu List* se chová v tomto případě jako FIFO, kdy se postupně nejstarší řešení zapomínají. V algoritmu (alg. 1) je také volně ponechána realizace ukončovací podmínky, která může být různá. Ukončovacím kritériem může být např. určitý počet kroků nebo absence změny x^* po určitém počtu kroků.

3.3.3.2 Aplikace Tabu Search v rozvrhovacím algoritmu

Pro Tabu Search jsou klíčové dvě proměnné – *Tabu List* a *Swap List*. Mějme krok optimalizace $step \in N$. Tabulka 3.4 zobrazuje rozdíly mezi těmito maticemi.

Tabulka 3.4: Vlastnosti proměnných *Tabu List* a *Swap List*.

Vlastnost	*Tabu List*	*Swap List*
význam	urychlení hledání *kandidátů*	ukončovací podmínka optimalizace, zpětná rekonstrukce rozvrhu
hodnoty	přesuny pro zisk kandidáta	reálné přesuny
dimenze	*tabuMemory* ×4, pracuje jako FIFO	dynamická ×4, FIFO bez vyprazdňování
četnost záznamu	vytvoření každého kandidáta	$Z_{next}^{step} < Z^{step}$
četnost čtení	vytváření každého kandidáta	nalezení max $\left(\frac{p_j}{dim_j} Z_j\right)$ v kroku $step + 1$
struktura řádku	index původní sestry, index nové sestry, index směny, *step*	

Optimalizace spočívá ve snížení hodnoty Z (provedení přesunu určité směny, obr. 3.3). Přesun by se měl uskutečnit tak, aby byl v daném okamžiku nejvýhodnější. Proto je nutné vyzkoušet všechny přípustné přesuny (v původním rozvrhu z kroku $step - 1$ se nalezne část rozvrhu, která produkuje největší příspěvek Z a zde se hledají kandidáti). Kandidátem nazýváme rozvrh, který vznikl přesunem směny od jedné sestry k druhé. Rozvrh kandidáta poté vstupuje do funkce pro výpočet cílové funkce Z. V případě, že

$$Z_{cand}^{step} < Z_{next}^{step} \quad , \tag{3.17}$$

je rozvrh kandidáta uložen do rozvrhu následníka. Následník je rozvrh, který se po dokončení kroku *step* za podmínky

$$Z_{next}^{step} < Z^{step} \quad , \tag{3.18}$$

uloží do rozvrhu stávajícího. V tomto případě je rozvrh stávající vstupem do kroku optimalizace $step + 1$ a přesun byl uložen na první řádek do matice *Swap List*. V opačném případě se hodnota Z a rozvrh nemění, přenáší se do kroku $step + 1$ a *Swap List* zůstává beze změny.

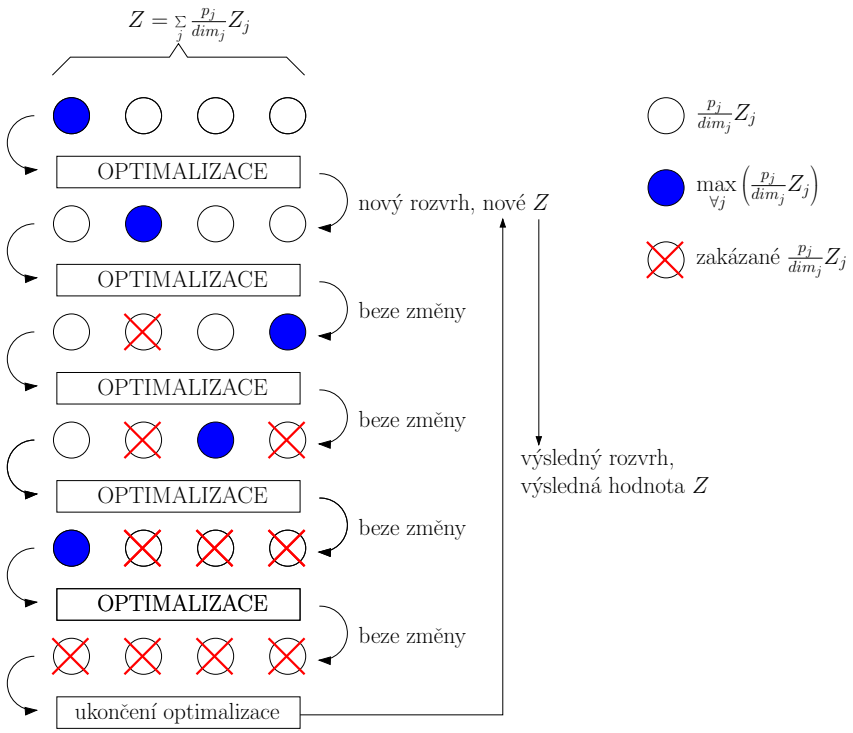
V následujícím kroku optimalizace $step + 1$ se hodnota kroku prvního řádku matice *Swap List* porovnává s hodnotou současného kroku ($step + 1$). Na základě tohoto porovnání je vytvořena ukončovací podmínka (více viz část 3.3.3.3).

Každého kandidáta je nutné vytvořit, zavolat výpočet Z a vyjádřit (3.17). Proto je na místě hledání kandidáta nějakým způsobem urychlit. V tom spočívá význam druhé

matice *Tabu List*. Při vytvoření prvního kandidáta se do matice zapíše hodnota přesunu pro zajištění, že algoritmus se už touto cestou při optimalizaci nevydá. Při vytváření dalších kandidátů se prochází *Tabu List*, zda-li v něm není totožný či opačný přesun. Pokud ano, přeskočí se k dalšímu kandidátovi. Pokud ne, provede se s kandidátem celý cyklus. Pro funkčnost metaheuristiky Tabu Search je zapotřebí přiměřený počet řádků (*tabuMemory* – konstanta zapomínání).

3.3.3.3 Ukončovací podmínka optimalizačního bloku

Konstrukce ukončovací podmínky souvisí se strukturou cílové funkce Z (více část 3.2). Hodnota cílové funkce Z je kritérium složené z podílů jednotlivých penalizací za nesplnění doplňujících podmínek. Pro představu konstrukce ukončovací podmínky optimalizačního bloku uvažujme cílovou funkci Z tvořenou čtyřmi složkami Z_i (obr. 3.4).



Obrázek 3.4: Konstrukce ukončovací podmínky.

V každém kroku optimalizace je snaha zmenšit podíl maximální složky Z. Pokud dojde ke zlepšení, optimalizuje se nový rozvrh s novou hodnotou cílové funkce Z. V druhém kroku optimalizace se nepodařilo zlepšit hodnotu Z, proto v dalším kroku probíhá optimalizace na složce Z, která má maximální podíl v Z z nezakázaných složek (při absenci

zákazů by došlo k zacyklení algoritmu). To se děje do té doby, dokud je v Z alespoň jedna nezakázaná složka. V posledním kroku je naznačena situace, kdy se nepodaří optimalizovat ani poslední přípustnou složku Z. V tomto okamžiku optimalizace končí a výsledný rozvrh je rozvrh z kroku optimalizace, kdy došlo k poslednímu zlepšení Z.

3.3.3.4 Parametry optimalizačního bloku

Během implementace optimalizačního bloku algoritmu bylo nutné sledovat určité parametry. Tyto parametry slouží k ladění algoritmu či jako informační zdroje při běhu algoritmu.

- zlepšení Z – po inicializaci počátečního rozvrhu je vždy hodnota funkce Z stejná (jedním z možných způsobů inicializace by také mohlo být náhodné generování přípustného rozvrhu, při kterém by výchozí hodnota Z vždy stejná nebyla), proto je vhodné sledovat zlepšení Z

$$\Delta Z = Z_0 - Z \tag{3.19}$$

- výpočetní čas algoritmu t_s – čas od zahájení inicializačního bloku po ukončení bloku výstupního (obr. 3.1)
- váhy penalizací p_j – více část 3.2.
- parametr metaheuristiky Tabu Search – při ladění metaheuristiky bylo také vhodné sledovat chování seznamu *Tabu List*. Zde figuruje parametr délky tohoto seznamu *tabuMemory* (konstanta zapomínání)
- provedené změny v rozvrhu – pro ilustraci provedených změn od počátečního rozvrhu do koncového rozvrhu slouží proměnná *Swap List*, jež zaznamenává uskutečněné přesuny směn (pomocí této proměnné lze rekonstruovat krok po kroku počáteční rozvrh)

3.3.4 Výstupní blok algoritmu

Vstupem do výstupního bloku algoritmu jsou konečné hodnoty rozvrhu, cílové funkce Z. Výstupem je graf vývoje cílové funkce Z a obrázek s konečným rozvrhem.

3.4 Pseudokód algoritmu heuristického řešení

Pro lepší pochopení předchozí části uveďme pseudokód algoritmu a také podrobnější blokové schéma, kde jsou naznačeny vztahy mezi jednotlivými funkcemi (obr. 3.5).

Algoritmus 2 NSPsolution (obr. 3.5, funkce 1) – spouštěcí funkce

Výstup: rozvrh pro m sester na n dní

begin

- **vstupní část**

  call inputData; {alg. 3, vstup dat, kontrola navrhnutelnosti}
  if ($nurse_lack$) **then**
    goto end; {ukončení návrhu rozvrhu}
  else
    goto inicializační část; {návrh rozvrhu}
  end if

- **inicializační část**

  call initialization; {alg. 4, tvorba počátečního rozvrhu}
  call consecShifts; {alg. 5, kontrola vytíženosti sester}
  call computeZ; {alg. 6, výpočet Z}
  goto optimalizační část;

- **optimalizační část**

  while $Z > 0$ **do**
    {výběr složky Z_j s největším podílem v Z, viz obr. 3.4}
    if (Z se nezměnilo) & (všechny $\frac{p_j}{dim_j} Z_j$ jsou zakázané) **then**
      goto výstupní část;
    else if (Z se nezměnilo) & ($\exists \frac{p_j}{dim_j} Z_j$ nezakázaná) **then**
      $constraint = \max\left(\frac{p_j}{dim_j} Z_j\right)$, $j \in$ nezakázané podmínky;
    else
      $constraint = \max\left(\frac{p_j}{dim_j} Z_j\right)$; {$Z$ se zlepšilo}
    end if

```
switch (constraint) do
  {optimalizuj podmínku constraint}
  case constraint :
    urči na základě constraint oblast rozvrhu pro nalezení kandidátů;
    v této oblasti prováděj přesuny pro nalezení následníka;
    for ∀ přípustné kandidáty na změnu rozvrhu do
      {hledání následníka}
      call changeShift;                    {alg. 7, výběr následníka}
    end for
  break;
end switch
call computeZ(rozvrh následníka);
Z := Z_next;                    {aktualizace rozvrhu a Z z následníka}
rozvrh := rozvrh následníka;
end while
```

- **výstupní část**

  vykresli graf závislosti Z na kroku optimalizace;

  vykresli rozvrh pro m sester na n dní;

end. {obr. 3.5, ukončení rozvrhovacího algoritmu}

Algoritmus 3 inputData (obr. 3.5, funkce 2) – zadání parametrů pro rozvrhování

Výstup: parametry pro rozvrhování – *nurse_lack*, $n, m, RE, RL, RN, weekday$

```
vlož n, m, RE, RL, RN, weekday;
{otestování navrhnutelnosti rozvrhu}
if (m ≤ (RE + RL + RN)) then
  nurse_lack := true;
  print "Rozvrh nelze navrhnout!";
else
  nurse_lack := false;
end if
```

Algoritmus 4 initialization (obr. 3.5, funkce 3) – generování počátečního rozvrhu

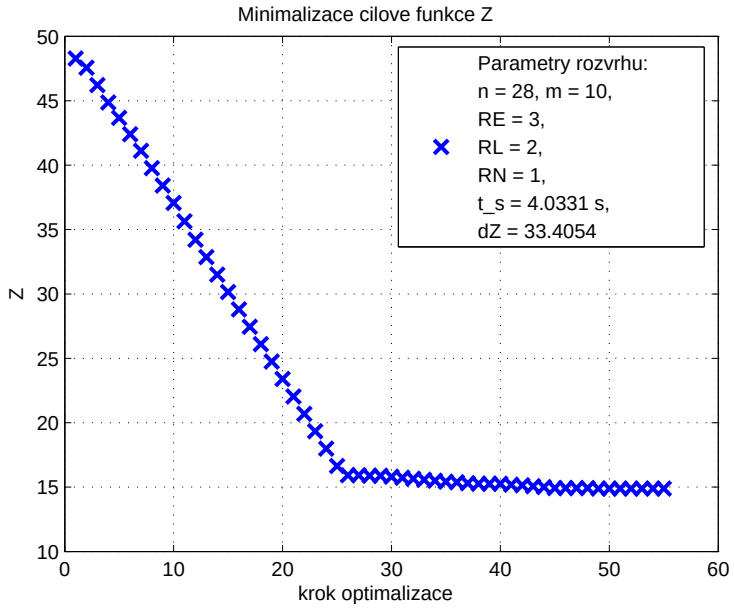
Vstup: parametry pro rozvrhování – $n, m, RE, RL, RN, weekday$

Výstup: *počáteční rozvrh*

```
for i=1:n do
  {obsazení směn na i-tý den dle požadavků RE, RL, RN}
  if (i > m) then
    k := i % m;                                   {přetečení indexu}
  else
    k := i;                                       {index v pořádku}
  end if
  {cyklické obsazení ranních směn – obr. 3.3}
  for early = 0 : RE − 1 do
    nurse = (day + early) % m;
    XE_{i,nurse} = 1;                             {ranní směna v rozvrhu XE}
    XS_{i,nurse} = 1;                             {ranní směna v celkovém rozvrhu}
  end for
  {cyklické obsazení odpoledních směn – obr. 3.3}
  for late = 0 : RL − 1 do
    nurse = (day + late + RE) % m;
    XL_{i,nurse} = 1;                             {odpolední směna v rozvrhu XL}
    XS_{i,nurse} = 2;                             {odpolední směna v celkovém rozvrhu}
  end for
  {cyklické obsazení nočních směn – obr. 3.3}
  for night = 0 : RN − 1 do
    nurse = (day + night + RE + RL) % m;
    XN_{i,nurse} = 1;                             {noční směna v rozvrhu XN}
    XS_{i,nurse} = 3;                             {noční směna v celkovém rozvrhu}
  end for
end for
```

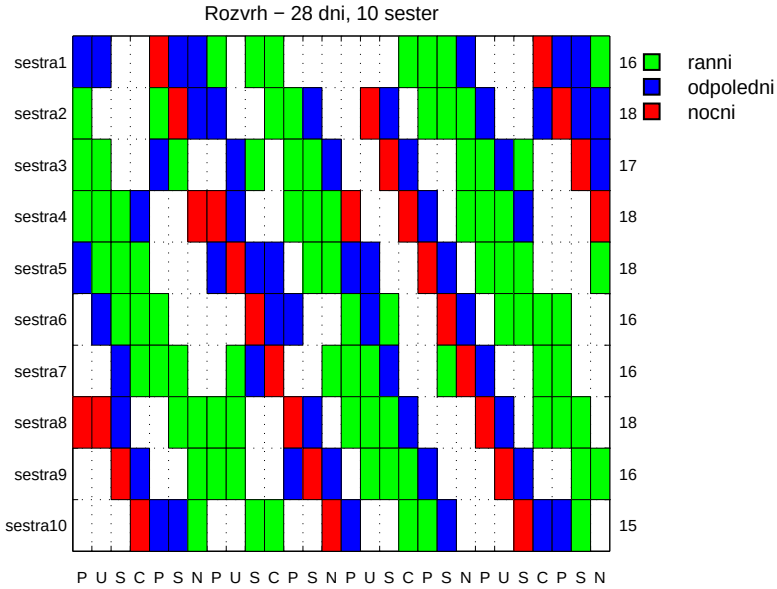
3.5 Výsledky

Pro ilustraci, jak si navržený algoritmus poradil s různými typy problémů, je zde uvedeno několik obrázků. Velmi důležité je, že během výpočtu odlišných problémů nebyly měněny váhy složek cílové funkce Z (netřeba podotýkat, že se změnou vah se mění i výpočetní čas, protože se mění i průběh výpočtu). Pokud by se tak stalo, bylo by irelevantní porovnávat hodnoty cílových funkcí Z. Hodnota této funkce je relativní, stejně tak jako zlepšení ΔZ, které sloužilo spíše pro orientaci při nastavování vah složek cílové funkce Z. Pro ukázku byly zvoleny čtyři různé příklady, z toho první tři jsou podobné složitosti a poslední je příklad náročnější na výpočetní čas.



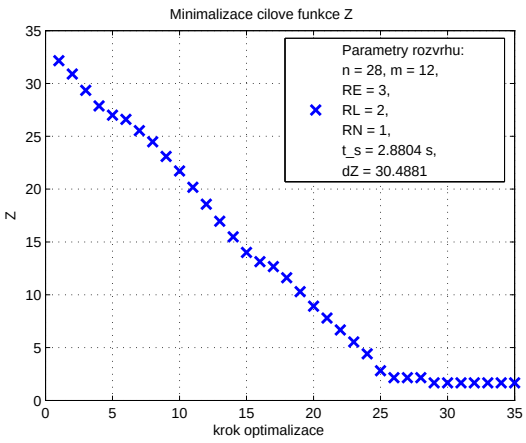
Obrázek 3.6: Optimalizace Z, příklad 1.

Na obr. 3.6 je zřejmá poměrně rychlá konvergence, která je způsobena optimalizaci největší složky Z. V druhé části grafu je naopak vidět optimalizace na složkách, které tak zřetelně do hodnoty funkce Z nepromlouvají.

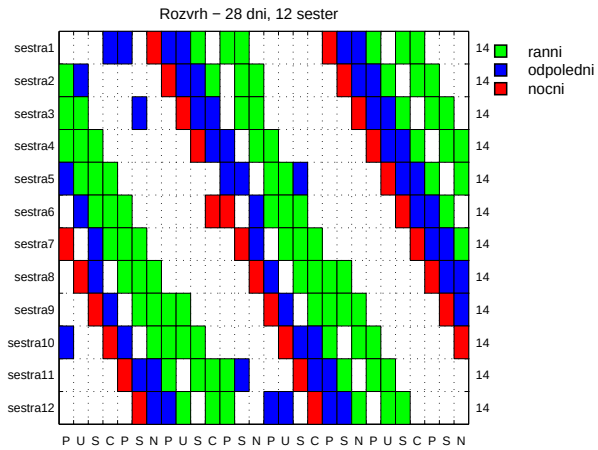


Obrázek 3.7: Výsledný rozvrh, příklad 1.

Pokud výsledný rozvrh (obr. 3.7) srovnáme s počátečním rozvrhem (obr. 3.2), lze vytušit souvislost s diagonální strukturou počátečního rozvrhu. Za povšimnutí v rozvrzích

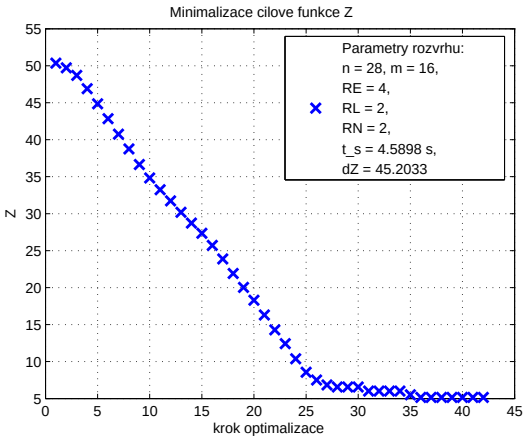


Obrázek 3.8: Optimalizace Z, příklad 2.

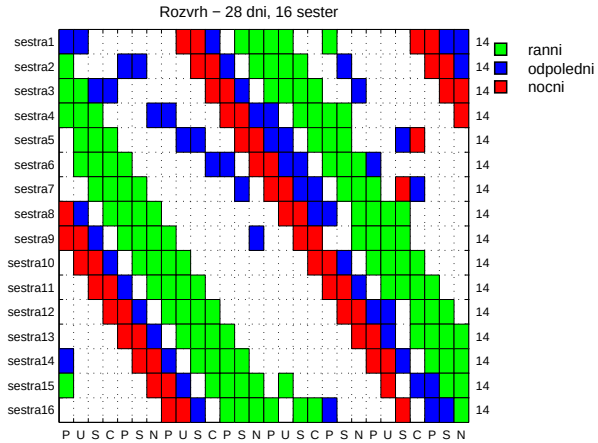


Obrázek 3.9: Výsledný rozvrh, příklad 2.

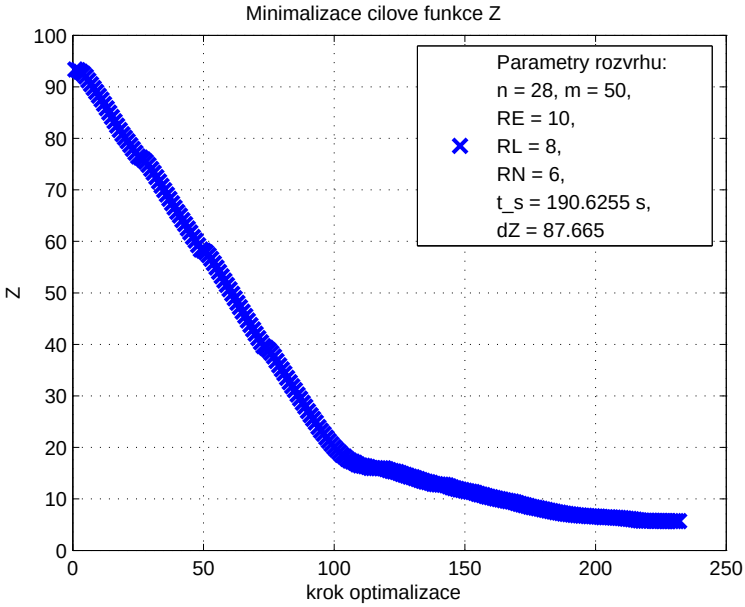
stojí, že nikde není porušena podmínka více jak 4 směn za sebou. Na tuto podmínku byl kladen velmi vysoký důraz. Grafy příkladů 2 a 3 jsou podobné či srovnatelné s grafem příkladu 1.



Obrázek 3.10: Optimalizace Z, příklad 3.

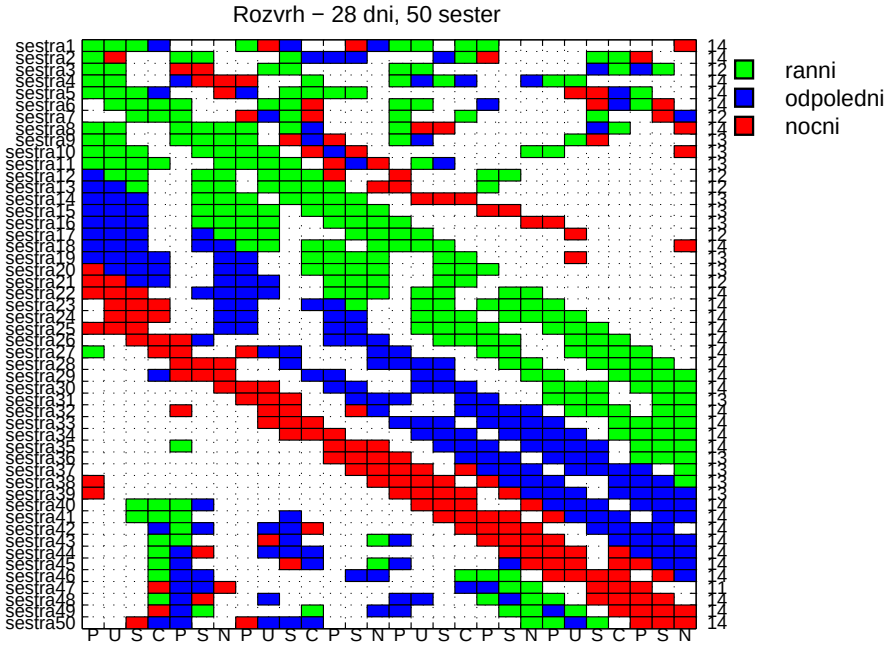


Obrázek 3.11: Výsledný rozvrh, příklad 3.



Obrázek 3.12: Optimalizace Z, příklad 4.

Poslední příklad ilustruje, jak obtížné úlohy (legenda obr. 3.12) je schopen algoritmus řešit v poměrně krátkém čase (ve srovnání s hledáním optimálního řešení z kapitoly 2).



Obrázek 3.13: Výsledný rozvrh, příklad 4.

3.6 Zhodnocení

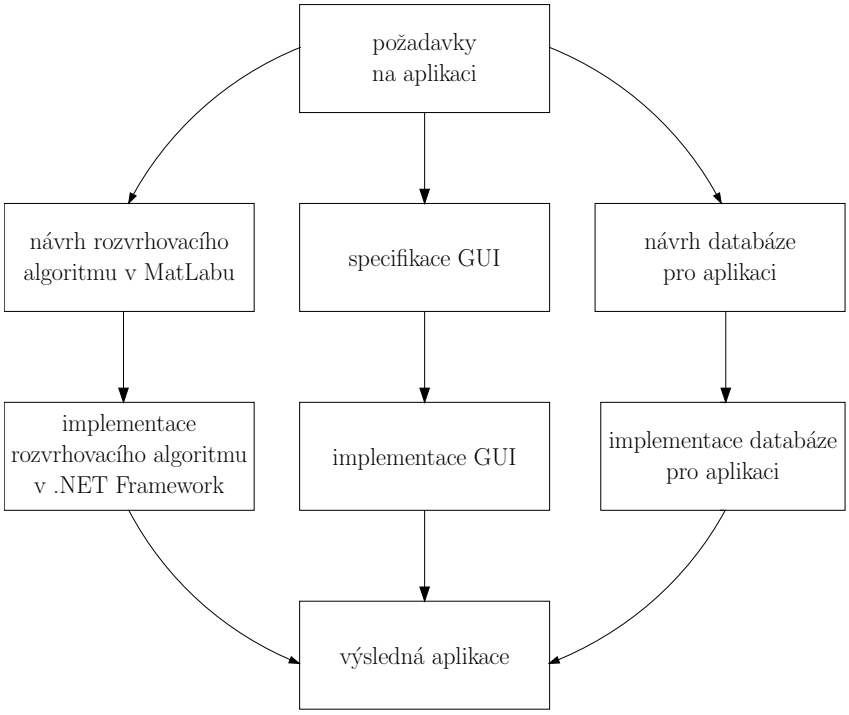
Algoritmus heuristického řešení pracuje uspokojivě. K řešení optimalizačního problému byla využita metaheuristika Tabu Search. Největším problémem při ladění algoritmu bylo správné nastavení vah p_j pro penalizace jednotlivých složek cílové funkce Z.

Algoritmus byl implementován v prostředí MatLab. Při vývoji aplikace sloužil tento výpočetní algoritmus jako hrubá kostra pro algoritmus vlastní aplikace. Implementace algoritmu proběhla v jiném programovacím jazyce, do implementace promluvila některá zlepšení (více kapitola 5, která se zabývá implementační částí aplikace).

Kapitola 4

Specifikace aplikace

Pro snadnější návrh aplikace je vhodné zodpovědět si ještě předtím, než s návrhem začneme, na otázku, co od aplikace čekáme, jak se bude chovat, pro koho je určená, jak by měla vypadat, jakou strukturu by měla mít atd. Souhrn těchto otázek můžeme nazvat jako specifikaci aplikace. Situaci vystihuje nejlépe obr. 4.1. V levé horní části je blok, kterému je věnována kapitola 3.



Obrázek 4.1: Blokové schéma návrhu aplikace.

Prostřední a pravá horní část odpovídá specifikaci aplikace, konkrétně návrhu GUI a

návrhu databáze, v které budou uložena data potřebná pro práci s aplikací (zaměstnanci a jejich údaje, rozvrhy jednotlivých oddělení). V neposlední řadě jsou součástí specifikace nároky na aplikaci zprostředkovaná nástroji jazyka UML.

4.1 Specifikace grafického rozhraní aplikace

Specifikace GUI[1] souvisí s funkčností aplikace, tzn. jaké jsou nároky na aplikaci, co od ní očekáváme a dále také vystihuje strukturu aplikace (logickou strukturu mezi jednotlivými částmi GUI).

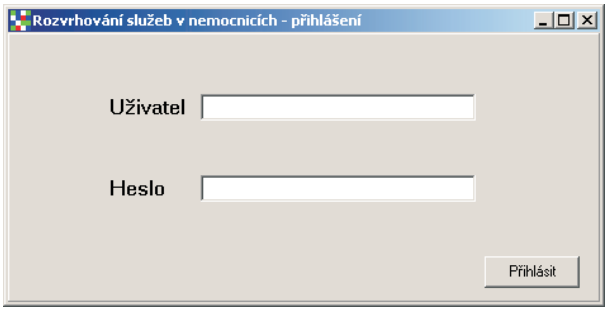
4.1.1 Základní myšlenky návrhu GUI

Aplikace pro rozvrhování služeb v nemocnici je z aplikačního hlediska informační (databázový) systém, kde data jsou generována rozvrhovacím algoritmem. Další data jsou nutná uchovávat k provozu aplikace samé (data zaměstnanců, různá nastavení atd.).

Dalším hlediskem je hledisko uživatelů, kteří budou aplikaci využívat. Vystupují zde dva typy uživatelských úrovní. První z nich je úroveň administrátora, tj. uživatele, který má práva pracovat s rozvrhem, s daty zaměstnanců (sester), měnit parametry návrhu rozvrhu a dělat ruční změny v rozvrhu. Roli administrátora může vykonávat např. hlavní sestra. Druhým typem uživatele je již řadový uživatel (jakákoliv sestra), který využívá aplikaci převážně pasivně, v režimu čtení. Mezi jejich práva patří prohlížení aktuálních rozvrhů a možnost ovlivnění vlastních směn a dnů volna na příští rozvrhovací období (nejčastěji v praxi jeden kalendářní měsíc). Jiným způsobem nemohou ovlivnit návrh rozvrhu.

Dvě výše zmíněná hlediska spojuje úvodní část aplikace, kterou je nepochybně přihlášení do systému na obr. 4.2.

[1]Specifikace GUI byla vyhotovena jako návrh, jak by mělo výsledné GUI, co se týče struktury a funkčních možností aplikace, vypadat. Samotná specifikace byla navržena v prostředí C++ Builder.



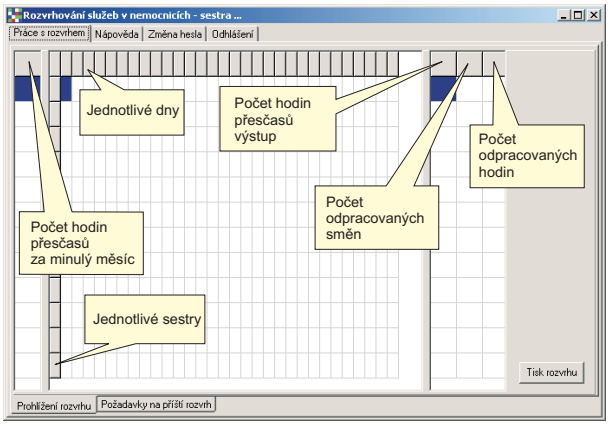
Obrázek 4.2: Přihlášení do aplikace.

4.1.2 Specifikace GUI pro řadového uživatele (sestru)

Po přihlášení se objeví několik záložek. První z nich je záložka Práce s rozvrhem.

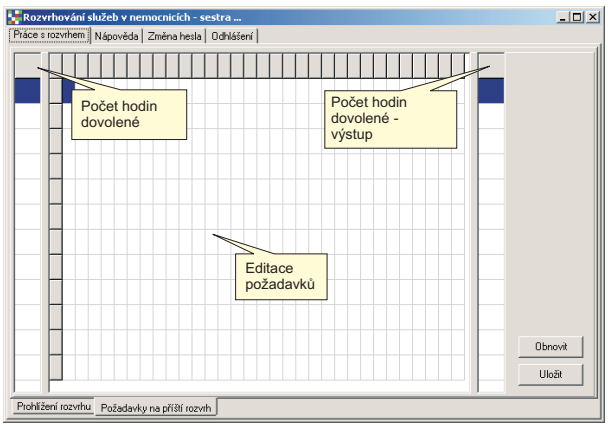
- Práce s rozvrhem

  Práce s rozvrhem má dvě podzáložky, první z nich je rozvrh na aktuální měsíc obr. 4.3. Tato obrazovka je pouze informativního rázu, tzn. pro čtení.



Obrázek 4.3: Rozvrh sester na aktuální měsíc.

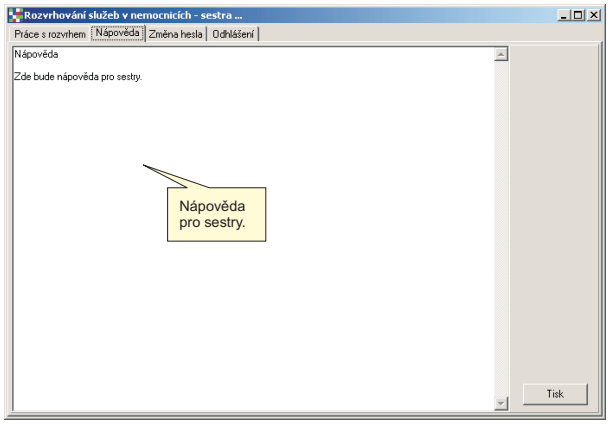
Další volbou je editace požadavků na příští měsíc (např. rezervace dovolené). Tato editace je vždy možná jen pro pole, která odpovídají přihlášené sestře, sestra nemůže editovat pole pro ostatní sestry.



Obrázek 4.4: Editace požadavků na příští měsíc.

- Nápověda

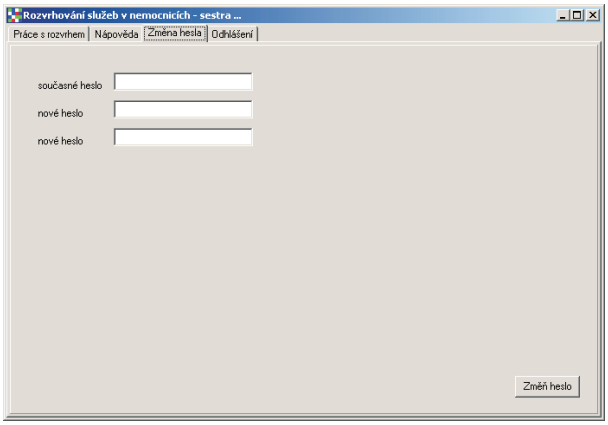
  Druhou záložkou je nápověda.



Obrázek 4.5: Nápověda pro sestry.

- Změna hesla

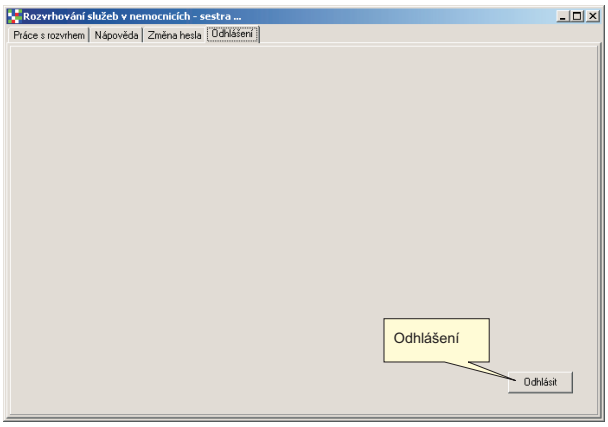
  Třetí záložka slouží ke změně hesla do systému.



Obrázek 4.6: Změna hesla uživatele.

- Odhlášení

  Poslední záložkou je odhlášení z aplikace.

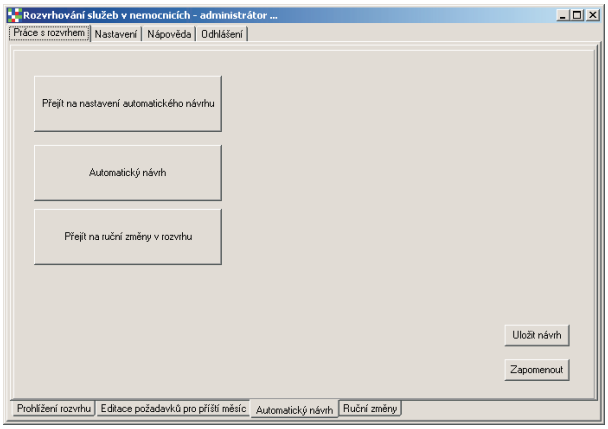


Obrázek 4.7: Odhlášení z aplikace.

4.1.3 Specifikace GUI pro administrátora

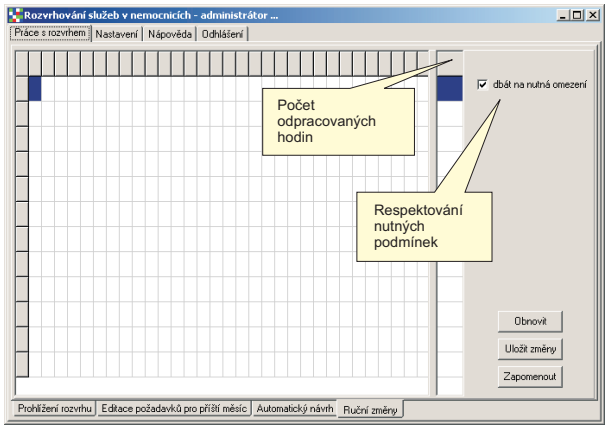
- Práce s rozvrhem

  Práce s rozvrhem pro administrátora má čtyři podzáložky, první dvě jsou až na mírné rozdíly stejné jako pro řadové uživatele. Administrátor má možnost zasahovat do požadavků *všech* sester na příští měsíc.



Obrázek 4.8: Automatický návrh rozvrhu.

Třetí podzáložka se zabývá automatickým návrhem rozvrhu na příští měsíc (obr. 4.8).

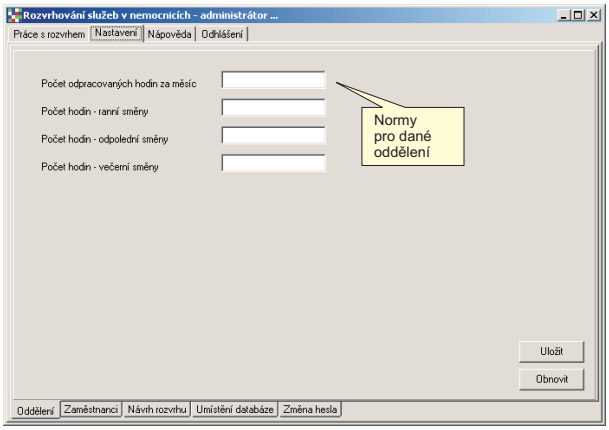


Obrázek 4.9: Ruční změny v automatickém návrhu.

Poslední podzáložka slouží k ruční editaci rozvrhu navrženého automatickým návrhem (obr. 4.9). Další záložkou administrátora je záložka s nastavením.

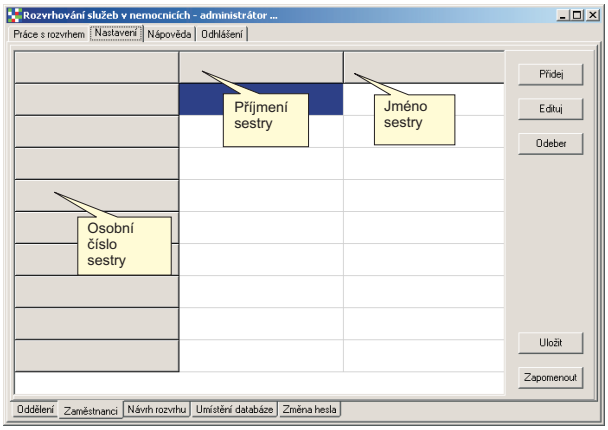
- Nastavení

  První z podzáložek je nastavení počtu normovaných hodin daného oddělení (obr. 4.10).

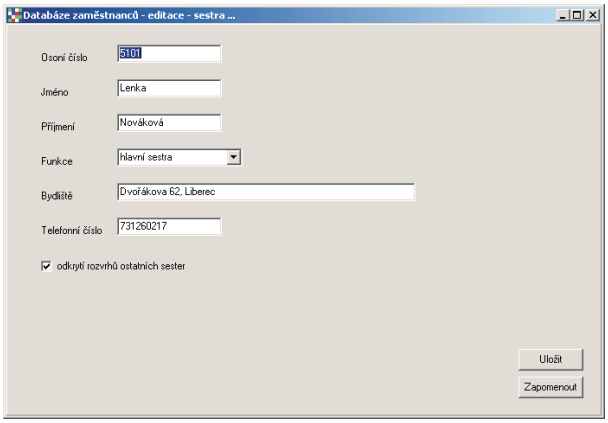


Obrázek 4.10: Nastavení údajů oddělení.

Druhou je editace databáze zaměstnanců (obr. 4.11), údaje o zaměstnancích jsou zobrazeny na (obr. 4.12).

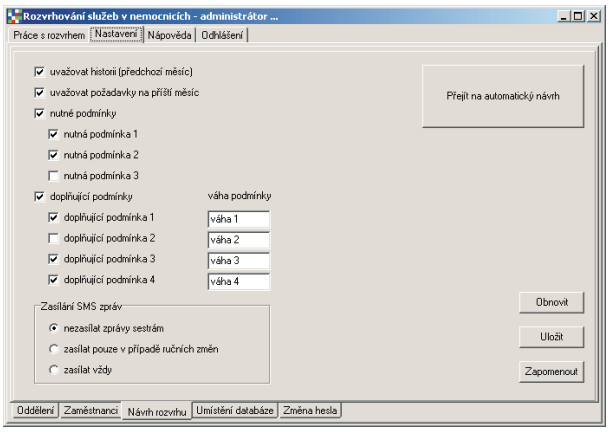


Obrázek 4.11: Databáze zaměstnanců.



Obrázek 4.12: Editace v databázi zaměstnanců.

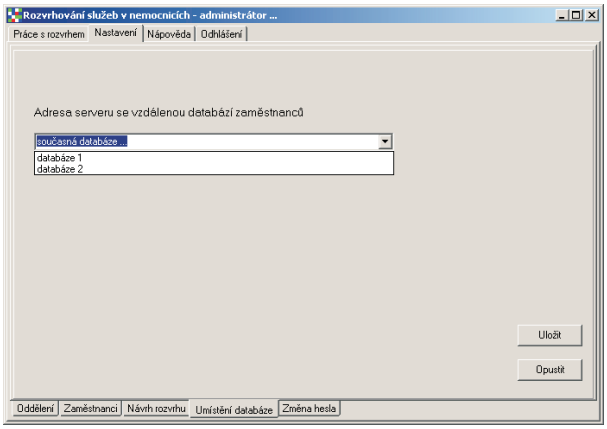
Třetí podzáložkou je nastavení parametrů pro automatický návrh rozvrhu (obr. 4.13). Kromě prvních dvou voleb, které jsou názvem výstižné, je zde možnost vybrat nutné



Obrázek 4.13: Parametry automatického návrhu.

a doplňující podmínky, které budou zahrnuty do automatického návrhu. U doplňujících podmínek lze nastavit váhu podmínky. Systém bude v budoucnosti zasílat SMS zprávy jednotlivým sestrám, budou zde tři různé úrovně pro upozornění na následující směnu (zasílat krátké textové zprávy vždy, zasílat pouze po ručních změnách, nezasílat nikdy).

Čtvrtá podzáložka slouží k nastavení adresy serveru, na kterém běží databáze zaměstnanců. Poslední podzáložka je změna hesla, totožná jako u řadových uživatelů (obr. 4.8).

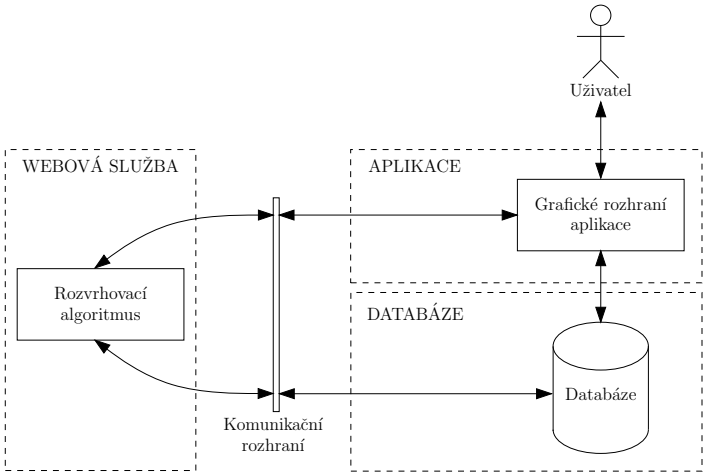


Obrázek 4.14: Nastavení databáze

- Nápověda, Odhlášení
  Tyto záložky jsou velmi podobné záložkám běžných uživatelů (obr. 4.5, obr. 4.7), nápověda se liší obsahem.

4.2 Návrh struktury aplikace

Tvorba aplikace je již nastíněna na obr. 4.1. Pro doplnění návrhu struktury aplikace je zde obr. 4.15. Rozvrhovací algoritmus je na aplikaci nezávislý. Uživatel přistupuje k apli-

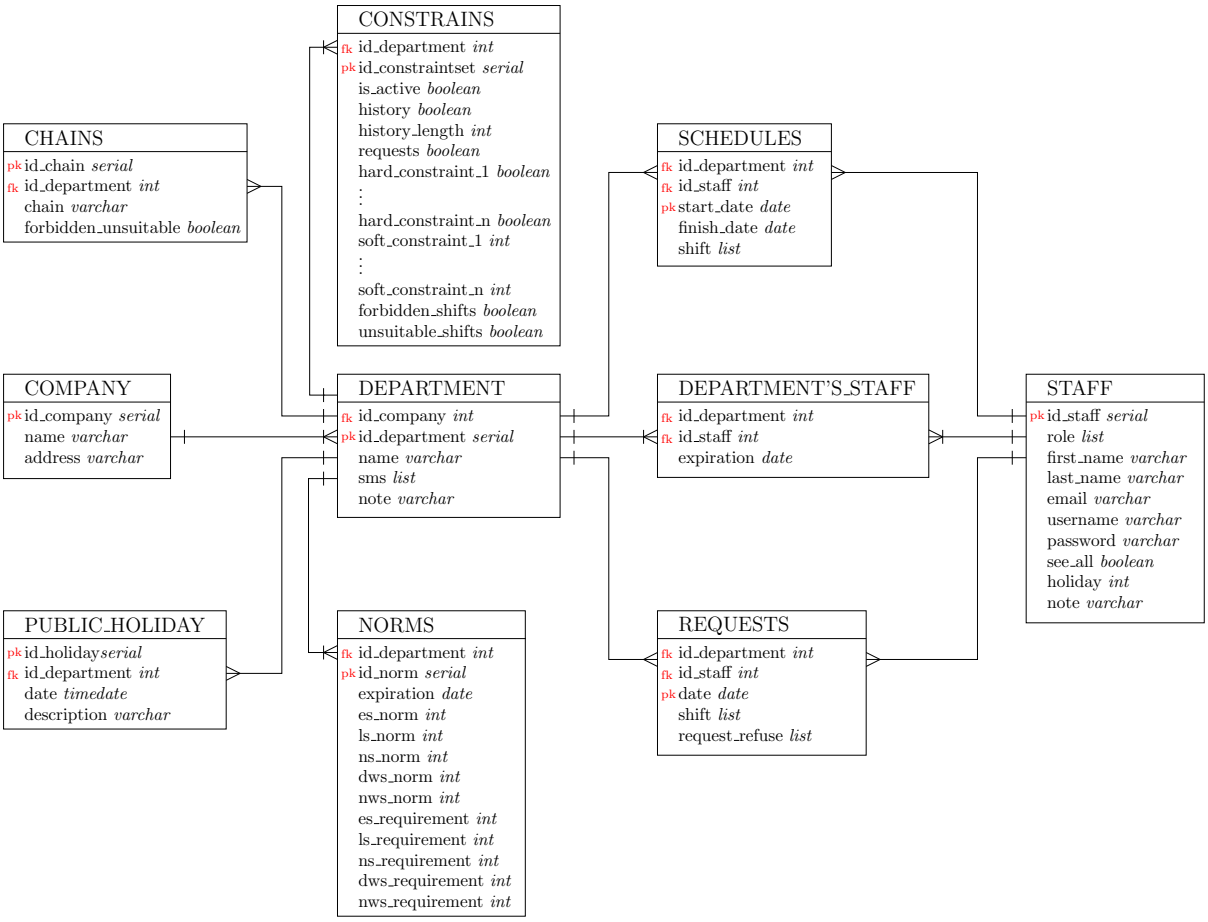


Obrázek 4.15: Struktura aplikace.

kaci prostřednictvím grafického rozhraní. Návrh rozvrhů provádí rozvrhovací algoritmus, potřebná data jsou ukládána v databázi. Následující řádky jsou věnovány struktuře databáze a specifikaci vlastní aplikace.

4.2.1 Návrh databáze, E-R diagram

Pro aplikaci byla navržena relační databáze. Návrh databáze je uveden v souladu s finální verzí aplikace (E-R diagram obsahuje některé datové položky, o kterých se více bude mluvit až v kapitole 5). Na obr. 4.16 je uveden E-R (Entity – Relation) diagram databáze s příslušnými položkami (tab. 4.2) a jejich datovými typy (tab. 4.1).



Obrázek 4.16: E-R diagram.

V E-R diagramu je několik tabulek, které jsou navzájem propojeny vazbami, většina z nich je typu $1 : n$. Jak je z diagramu vidět, v jednom oddělení je jedna až n sester, ale také jedna sestra může pracovat ve více jak jednom oddělení. Tabulek s normami (NORMS) pro dané oddělení muže být víc (z důvodu budoucího napojení na mzdovou část aplikace je nutné uchovávat v databázi i minulé nastavení norem), opět je zde vazba $1 : n$. Do tabulky se svátky (PUBLIC_HOLIDAY) se ukládají data týkající se dnů pracovního klidu. Tabulka CHAINS uchovává u každého oddělení nepřípustné a neoblíbené kombinace.

Tabulka 4.1: Význam použitých výrazů v E-R diagramu, datové typy.

Výraz	Význam
pk	primární klíč
fk	cizí klíč
serial	datový typ pro identifikace, celočíselný, jedinečný, inkrementálně se zvětšující
varchar	řetězce proměnné délky
list	výčtový typ
int	celočíselné hodnoty
boolean	binární datový typ (true, false)
date	datový typ pro data

4.2.2 Objektově orientovaná analýza aplikace

Pro analýzu určitého systému máme dvě možnosti. První z nich je *objektově orientovaná analýza*, která je založena na generalizaci. Začíná se z konkrétní části systému, která se analyzuje a popíše. V dalších částech se pak hledají společné rysy, z nichž se odvozují obecné zákonitosti. Výhodou je možnost rychlého paralelního vývoje prototypového software, soulad s moderními prostředky softwarového vývoje. Do objektově orientované analýzy spadá velké množství typů modelů, které jsou deklarovány v jazyku UML [27] (Unified Modeling Language). V našem případě nám postačí diagram užití (model požadavků, Use Case Diagram) korespondující se specifikací GUI z předchozí části.

Tabulka 4.2: Datové položky v E-R diagramu.

Tabulka	Datová položka	Význam	Příklad
COMPANY	id_company	identifikace společnosti	001
	name	název společnosti	ÚVN
	address	adresa společnosti	Praha...
DEPARTMENT	id_company	identifikace společnosti	001
	id_department	identifikace oddělení	01
	name	název oddělení	Chirurgie
	sms	zasílání sms zaměstnancům	nezasílat[2]
	note	poznámka k oddělení	poznámka
CONSTRAINTS	id_department	identifikace oddělení	01
	id_constraintset	identifikace sady parametrů návrhu	01
	is_active	příznak, zda-li je sada parametrů návrhu aktivní	true
	history	příznak, zda-li brát při návrhu rozvrhu v potaz historii	true
	h_length	počet dní, na který se má brát v úvahu historie rozvrhu	5
	requests	příznak, zda-li brát při návrhu rozvrhu v potaz požadavky sester	true
	hard_constraint_i	zahrnovat i-té nutné omezení do návrhu rozvrhu	true
	soft_constraint_i	zahrnovat i-té doplňující omezení do návrhu rozvrhu s vahou	30
	forbidden_shifts	příznak, zda-li brát v úvahu nepřípustné směny	true
	unsuitable_shifts	příznak, zda-li brát v úvahu neoblíbené směny	true
NORMS	id_department	identifikace oddělení	01
	id_norm	identifikace normy	01
	expiration	datum platnosti norem pro dané oddělení	30-06-05[3]
	es_norm	norma ranní směny v hodinách	8
	ls_norm	norma odpolední směny	8
	ns_norm	norma noční směny	8
	dws_norm	norma víkendové denní směny	12
	nws_norm	norma víkendové noční směny	12
	es_requirements	požadavek na počet zaměstnanců na ranní směně	4
	ls_requirements	požadavek na počet zaměstnanců na odpolední směně	3
	ns_requirements	požadavek na počet zaměstnanců na noční směně	2
	dws_requirements	požadavek na počet zaměstnanců na víkendové denní směně	2
	nws_requirements	požadavek na počet zaměstnanců na víkendové noční směně	1
CHAINS	id_chains	identifikace řetězce s nepřípustnýmineoblíbenými směnami	1
	id_department	identifikace oddělení	1
	chain	hodnota nepřípustnéneoblíbené směny	RNRN
	forbidden_unsuitable	příznak, o jaký typ řetězce se jedná	true[4]
PUBLIC_HOLIDAY	id_holiday	identifikace svátku	10
	id_department	identifikace oddělení	1
	date	datum svátku, dnu pracovního klidu	1-1-2006, 0:00[5]
	description	popis svátku	Nový rok
DEPARTMENT'S STAFF	id_department	identifikace oddělení	01
	id_staff	identifikace zaměstnance	20
	expiration	datum platnosti, kdy patřil zaměstnanec do daného oddělení	30-06-05[3]
SCHEDULES	id_department	identifikace oddělení	01
	id_staff	identifikace zaměstnance	20
	start_date	datum a čas počátku dané směny	29-05-05, 14:00
	finish_date	datum a čas konce dané směny	29-05-05, 22:00
	shift	typ směny	R (ranní)
REQUESTS	id_department	identifikace oddělení	01
	id_staff	identifikace zaměstnance	20
	start_date	datum a čas počátku požadavku	29-05-05, 00:00
	finish_date	datum a čas konce požadavku	29-05-05, 23:59
	shift	typ směny	D (dovolená)
	request_refuse	požadovaná/nechtěná směna	požadovaná

[2]Datová položka pro zasílání sms zpráv může nabývat hodnot nezasílat sms, zasílat pouze v případě ručních změn, zasílat vždy; v případě funkcí sester je výběr z funkcí hlavní sestra, zaučená sestra, pomocná sestra.

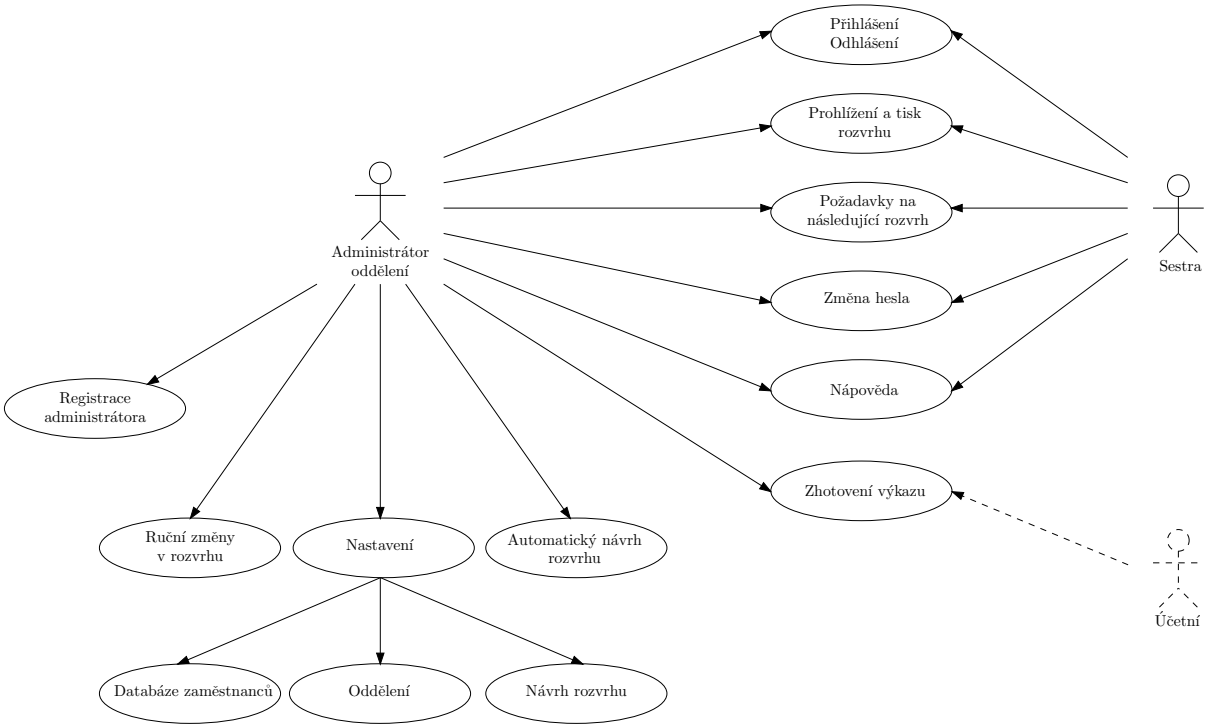
[3]Data mohou být buď ukončená (např. 30-06-05), anebo současná, tzn. daný stav ještě trvá.

[4]V případě, že je příznak roven hodnotě true, jedná se o řetězec nepřípustné kombinace směn, v opačném případě jde o neoblíbenou kombinaci směn.

[5]Datum se je v databázi z důvodu mzdové části aplikace pro zpětnou rekonstrukci údajů v celém tvaru, v aplikaci se však pracuje pouze s dnem a měsícem svátku.

Tabulka	Datová položka	Význam	Příklad
STAFF	id_staff	identifikace zaměstnance	20
	role	funkce zaměstnance	1 (hlavní sestra)[2]
	first_name	jméno zaměstnance	Lenka
	last_name	příjmení zaměstnance	Nováková
	email	email zaměstnance	novakl@seznam.cz
	username	uživatelské jméno pro vstup do aplikace	novakl
	password	heslo pro vstup do aplikace	mojeheslo
	see_all	parametr pro viditelnost rozvrhu ostatních sester	true
	holiday	počet hodin dovolené	88
	note	poznámka o zaměstnanci	snaživá

Diagram užití (obr. 4.17) je model popisující možné způsoby použití navrhovaného systému (entity v oválech), včetně příslušných typů uživatelů (aktérů – osoby nebo činitelé mimo systém, jež se systémem interagují). Jedná se o vnější popis aplikace bez specifikace její vnitřní struktury. Na první pohled jsou zřejmé funkce aplikace a odlišná práva pro různé uživatele. U entity *Zhotovení výkazu* by mohl být napojen další aktér, např. účetní.



Obrázek 4.17: Diagram užití (Use Case Diagram).

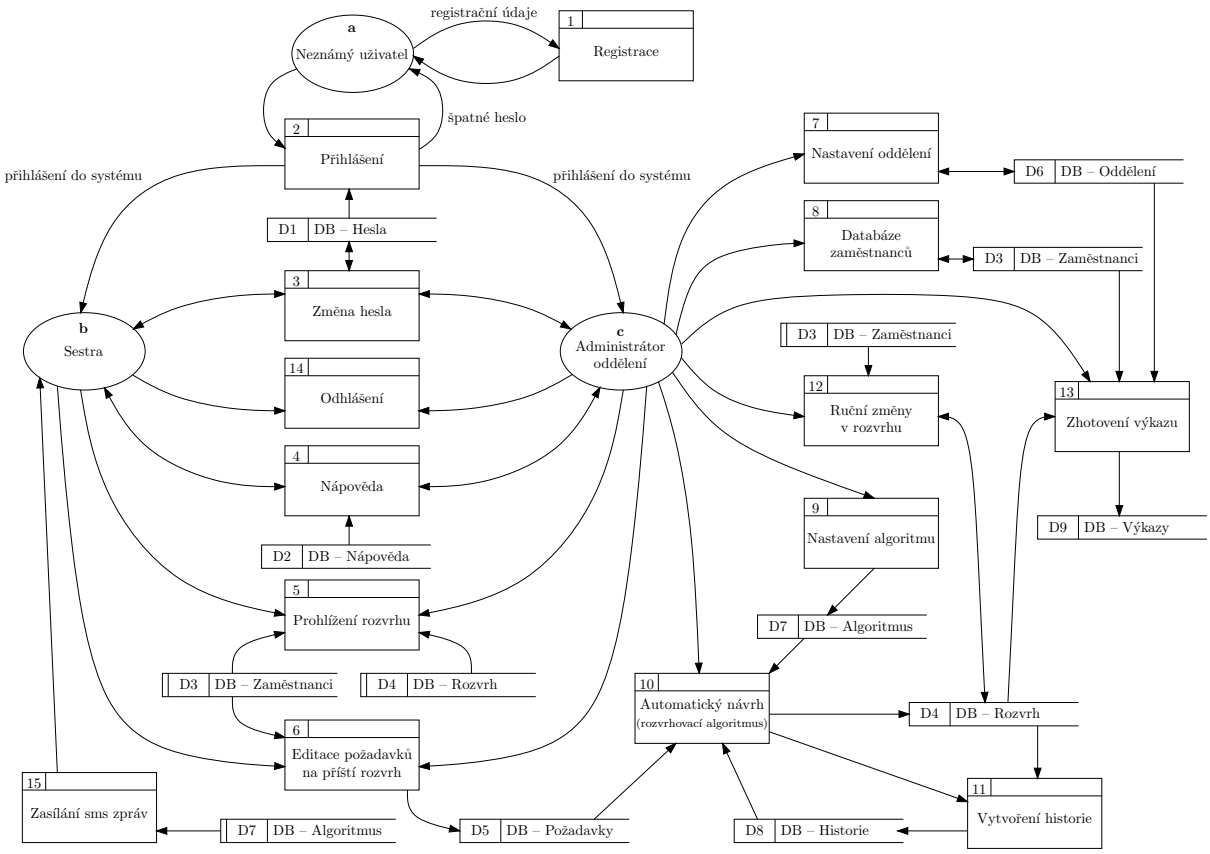
4.2.3 Strukturovaná analýza aplikace

Opačným přístupem je *strukturovaná analýza* založená na specifikaci. Chápe systém jako celek a postupně ho dekomponuje na jednodušší části. Je metodicky velmi dobře zvládnuta. Nevýhodou je absence možnosti ukončit předčasně analýzu a implementovat konkrétní část systému. Typickým příkladem strukturované analýzy je metoda SSADM (Structured Systems Analysis and Design Method). Tato metoda kombinuje tři základní techniky (modelování logických datových struktur, modelování datových toků, modelování chování entit). Pojem strukturovaná analýza pokrývá velmi velký rozsah, v našem případě si vystačíme pouze s modelováním datových toků (Data Flow Diagram, Data Flow Modeling). Diagram datových toků v aplikaci je zobrazen na (obr. 4.18).

Pro diagram datových toků je zavedena následující syntaxe.

- ovály v modelu jsou externí entity (*Neznámý uživatel, Sestra, Administrátor oddělení*), značí se malým písmenem
- obdélníky se značí procesy, jejichž identifikátor je číslo (seřazeno chronologicky a dále podle důležitosti procesu)
- otevřené obdélníky symbolizují datové sklady (jakékoliv úložiště dat, ne nutně databáze), značí se D + číslo
- šipky jsou datové toky mezi entitami, procesy a datovými sklady
- duplikované entitě či datovému skladu se přidává pruh

Z modelu je zřejmé, že do systému nejprve vstupuje *Neznámý uživatel.* V závislosti na správnosti hesla mu je ponechán statut *Neznámého uživatele* nebo změněn na entity *Sestra* či *Administrátor oddělení.* Obě tyto entity, přihlášené do systému, mají přístup k určitým procesům, při komunikaci mezi nimi proudí data. Z diagramu jsou velmi dobře viditelné závislosti procesů i entit (uživatelů) na databázi, ve které jsou data uložena.



Obrázek 4.18: Diagram datových toků (Data Flow Diagram).

4.3 Zhodnocení

V této kapitole byla navržena specifikace grafického rozhraní aplikace a dále provedena analýza struktury a specifikace vlastní aplikace. Během implementace aplikace se vycházelo z obsahu této kapitoly.

Kapitola 6

Závěr

Ústředním motivem této práce byl pojem rozvrhování. Cílem byl návrh aplikace pro rozvrhování služeb v nemocnicích. Aplikace byla navrhnuta jako webová, což s sebou přináší nesporné výhody (přístup k aplikaci přes webové rozhraní, absence instalace aplikace, nezávislost na operačním systému, zadávání požadavků sester přímo do aplikace). Další výhodou je při implementaci kladený důraz na intuitivnost a přehlednost ovládání.

Klíčová pro návrh aplikace byla implementace rozvrhovacího algoritmu. Cesta k výsledné aplikaci vedla přes studium rozvrhovacích algoritmů. Nejprve bylo nalezeno optimální řešení problému, které však bylo v praxi nepoužitelné z důvodu velké časové náročnosti. Proto byl implementován vlastní rozvrhovací algoritmus spadající do oblasti vícekriteriálního programování v kombinaci s metaheuristikou Tabu Search. Tato cesta je kompromisem mezi kvalitou řešení a časem potřebným k rozvrhování. Uživatel má možnost výsledný rozvrh korigovat v podobě ručních změn, případně se rozhodnout po korekci k další iteraci návrhu rozvrhu.

Aplikace má ryze praktické uplatnění, je vyvíjena ve spolupráci s Psychiatrickou léčebnou Bohnice, kde bude v budoucnosti nasazena. Aplikace i rozvrhovací algoritmus samotný je možné iterativně zdokonalovat dle zkušeností koncových uživatelů ku jejich spokojenosti.

Literatura

[1] Hung, R.: *Hospital Nurse Scheduling*, Journal of Nursing Administration, Volume 25, number 78, 1995.

[2] Cheang, B.; Li, B.; Lim, A.; Rodrigues, B.: *Nurse Rostering Problems A Bibliographic Survey*, European Journal of Operations Research, 2002.

[3] Eiselt, H. A.; Sandblom, C. L.: *Integer Programming and Network Models*, Springer-Verlag Berlin and Heidelberg, 2000.

[4] Abernathy, W. J.; Baloff, N.; Hershey, J. C.; Wandel, S.: *A Three-Stage Manpower Planning And Scheduling Model*, Operations Research 22, 1973.

[5] Warner, M.; Prawda, J.: *A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital*, Management Sience 19, 1972.

[6] Trivedi, V., M.; Warner, M.: *A Branch and Bound Algorithm for Optimum Allocation of Float Nurses*, Management Sience 22, No. 9, 1976.

[7] Arthur, J.; Ravindran, A.: *A Multiple Objective Nurse Scheduling Model*, AIIE Transactions, vol. 13, no. 1, 1981.

[8] Franz, L., S.; Baker, H., M.; Leong, G. K.; Rakes, T. R.: *Mathematical Model for Scheduling and Staffing Multiclinical Health Regions*, European Journal of Operations Research, vol. 41, no. 3, 1989.

[9] Okada, M.; Okada, M.: *Prolog-Based System for Nursing Staff Scheduling Implemented on a Personal Computer*, Computers and Biomedical Research 21, 1988.

[10] Okada, M.: *An Approach to the Generalised nurse Scheduling Problem – Generation of a Declarative Program to Represent Institution-Specific Knowledge*, Computers and Biomedical Research 25, 1992.

[11] Weil, G.; Heus, K.; Francios, P.: *Constraint Programming for Nurse Scheduling*, IEEE Engineering in Medicine and Biology, 1995.

[12] Meisels, A.; Gudes, E.; Solotorevski, G.: *Employee Timetabling, Constraint Networks and Knowledge-Based Rules*, First International Conference Edinburgh, Springer, 1995.

[13] Smith, L., D.; Bird, D.; Wiggins, A.: *A Computerised System to Schedule Nurses that Recognises Staff Preferences*, Hospital & Health Service Administration, 1979.

[14] Chen, J., G.; Yeung, T.: *Hybrid Expert System Approach to Nurse Scheduling*, Computers in Nursing, 1993.

[15] Berrada, I.; Ferland, J.; Michelon, P.: *A Multi-Objective Approach To Nurse Shceduling with both Hard and Soft Constraints*, Socio-Economic Planning Science 30, 1996.

[16] Dowsland, K.: *Nurse Scheduling with Tabu Search and Strattegic Oscillation*, European Journal of Operations Research 106, 1998.

[17] Berghe, G. V.: *An Advanced Model and Novel Meta-Heuristic Solution Methods to Personnel Scheduling in Healthcare*, PhD dissertation, University of Gent, 2002.

[18] Easton, F.; Mansour, N.: *A Distributed Genetic Algorithm for emplyee Staffing and Scheduling Problems*, Conference on Genetic Algorithms, San Mateo, 1993.

[19] Aickelin, U.: *Genetic Algorithms for Multiple-Choice Optimisation Algorithms*, PhD thesis, European Business Management School University of Swansea, 1999.

[20] [online] *Demo produkt The Nurse Scheduling System*, společnost Harborsoft, dostupný na <http://www.harborsoft.com/schdemo8.exe>.

[21] [online] *Produkt NSPGen*, Univerzita v Ghentu, fakulta ekonomická, dostupný na <http://www.projectmanagement.ugent.be/nsp.php>.

[22] [online] *Produkt Time Tracker*, společnost Asgard Systems, dostupný na <http://www.asgardsystems.com/download/ttsystem.exe>.

[23] Studio L: *sL-Plán – návod k obsluze*, Česká Lípa, 1992.

[24] [online] *Dokumentace nástroje GLPK*, dostupná na <http://www.gnu.org/software/glpk/glpk.html#TOCdocumentation>.

[25] Azaiez, M. N.; Al Sharif, S. S.: *A 0-1 goal programming nodel for nurse scheduling*, Computers & Operations Research 32, 2005.

[26] Pelikán, J.: *Diskrétní modely v operačním výzkumu*, Professional Publishing, 2001.

[27] Arlow, J.; Neustadt, I.: *UML a unifikovaný proces vývoje aplikací*, Computer Press, 2002.

[28] Jalloul, G.: *UML by Example*, Cambridge University Press, 2003.

[29] Kačmář, D.: *Programujeme .NET aplikace ve Visual Studiu .NET*, Computer Press, 2001.

[30] Prosise, J.: *Programování v Microsoft .NET*, Computer Press, 2003.