



SEMESTRÁLNÍ PRÁCE Z PŘEDMĚTU UZI
ZNALOSTNÍ SYSTÉM - LETIŠTĚ

1 Formulace úlohy

Je dána množina typů letadel a pro každý typ letadla je dán seznam stojánek, u kterých může stát. Na každé stojánce může v daný čas stát pouze jedno letadlo, s výjimkou stojánky "volná plocha", na které může stát libovolně mnoho letadel (existují ale letadla, která nemohou stát na volné ploše – např. Concorde). Dále je dán letový plán, tj. seznam letadel s určením jejich typu, doby příletu a odletu. Cílem je tedy vytvořit program ve formě znalostního systému, který rozvrhne letadla na jednotlivé stojánky tak, aby co nejvíce letadel bylo obhospodařeno stojánkami (tj. ne na volné ploše). Letiště je definováno množinou všech stojánek a volnou plochou s daným počtem míst.

Takovéto zadání lze interpretovat tak, že vstupem programu budou dvě sady dat. První sada budou definice letiště (stojánky a počet volných stání) a jednotlivých letadel a druhá sada, která bude reprezentovat letový plán. Výstupem této úlohy by tedy mělo být korektní rozdělení všech letů z letového plánu na jednotlivé stojánky. Vstupní data mohou vypadat například takto:

airport:

stojanek1

stojanek2

stojanek3

stojanek4

stojanek5

free 20

aircraft:

Airbus A320;s1;s3

Boeing 737;s2

Concorde;s5;s8;s12;x

Airbus A380;s12;s13;s14;s15

Takováto data by obsahovala právě první zmíněná sada dat, respektive blok, který začne klíčovým slovem **airport:** bude představovat inicializační parametry pro letiště (ve výše uvedeném případě tři stojánky a volná plocha s dvaceti místy). Druhý blok uvozený klíčovým slovem **aircraft:** naopak definuje jednotlivá letadla, se kterými dokáže letiště pracovat. V tomto případě jsou definována dvě letadla, ve formátu, kde první řádka představuje název letadla a druhá jednotlivé stojánky, na kterých může letadlo stát. Pokud výčet stojánek končí znakem "x", znamená to, že letadlo nemůže na letišti využít volnou plochu. Druhá datová sada specifikuje již zmíněný letový plán. Ten vychází z dat definovaných v první datové sadě. Pokud nebude odpovídat první datové sadě (např. let s nedefinovaným letadlem), budou daná data nevyužita. Data, která popisují letový plán, mohou vypadat následovně:

Airbus A320

08:00;10:00

11:30;13:30

-

Boeing 737

12:00;21:00

14:30;15:30

-

Concorde

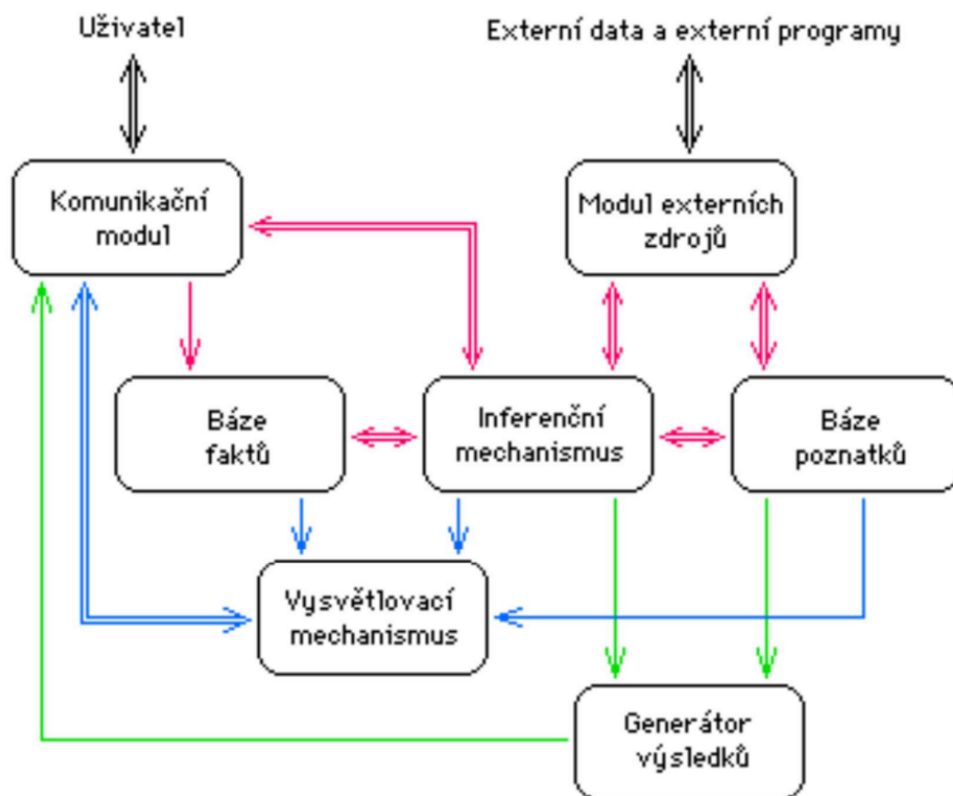
04:00;06:00

11:30;13:30

Takto uvedená data specifikují jednotlivé lety pro určité letadlo. V tomto případě bude dané letadlo mít dva lety, které jsou uvedené ve formátu **přílet;odlet**. Pro jednoduchost je program koncipován tak, že se celý letový plán odehrává v jeden den, tedy možné časy odletů a příletů začínají v 00:00 a končí v 23:59. Takovéto omezení nedovoluje uživateli zadat čas příletu například 23:30 a odlet až následující den, například 00:30.

2 Analýza úlohy

Vzhledem k povaze úlohy, tedy že se má jednat o znalostní systém, je nutné zajistit, aby úloha naplňovala základní vlastnosti takového systému. Program by tedy měl splňovat svou formou a strukturou konvence návrhu znalostních systémů. Architektura znalostního systému může vypadat následovně:



Obrázek 1: Příklad architektury znalostního systému.

Jedním z cílů tedy bude správně dekomponovat aplikaci do jednotlivých modulů a efektivně je mezi sebou provázet. V kontextu této práce tedy inferenční modul bude obstarávat korektní běh celého systému, respektive jeho úkolem bude správně využívat ostatní moduly za účelem plynulé obsluhy systému. Báze poznatků (znalostí) bude v tomto systému definována již zmíněnými sadami dat (viz kapitola 1). Báze faktů bude klíčová pro rozvrhnutí letadel na jednotlivé stojánky. Bude obsahovat struktury a metody pro efektivní průběh rozvrhnutí obsazení stojánek. V kooperaci s generátorem výsledků tedy vytvoří odpovídající plán obsazení. Vzhledem k tomu, že se jedná o znalostní systém, bude klíčové poskytnout uživateli vysvětlení, proč systém provedl rozvržení daným způsobem. Tuto problematiku by měl řešit vysvětlovací modul. V neposlední řadě samotnou komunikaci s uživatelem (případně vnějšími soubory) bude obstarávat komunikační modul.

3 Popis algoritmu řešení

3.1 Možnosti řešení

Tato úloha představuje problém v informatice definovaný jako přiřazení a plánování zdrojů, kde cílem je optimálně přiřadit letadla ke stojánkům tak, aby co nejvíce letadel bylo umístěno na dedikovaných stojáncích (a ne na volné ploše). Tento typ úloh nabízí několik způsobů řešení, různé časové a výpočetní náročnosti.

Jedním z možných přístupů je určitá forma **greedy** algoritmu. Jedná se o jednoduchou a intuitivní metodu, která se rozhoduje na základě lokálně optimálních voleb v každém kroku s cílem dosáhnout globálně optimálního řešení. V kontextu problému rozvržení letadel na stojánky tento algoritmus zpracovává letadla jedno po druhém, seřazená například podle času příletu. Každé letadlo se pokusí přiřadit k první volné dedikované stojánce, která splňuje jeho požadavky na typ letadla a časový interval, kdy bude na letišti. Pokud žádná vhodná stojánka není volná, algoritmus přiřadí letadlo na volnou plochu. Tento přístup je rychlý, protože provádí rozhodnutí okamžitě na základě aktuálního stavu bez nutnosti zkoumat všechny možné budoucí scénáře. Greedy algoritmus

je snadno implementovatelný a výpočetně nenáročný, což ho činí vhodným pro malé až středně velké instance problému. Nicméně jeho nevýhodou je, že nemusí vždy najít globálně optimální řešení, protože se rozhoduje pouze na základě okamžitých podmínek, což může vést k suboptimálním výsledkům, zejména pokud se nebere v úvahu dlouhodobý dopad jednotlivých rozhodnutí.

Další možností je heuristický přístup, který se snaží najít řešení problému pomocí pravidel a odhadů, které pomáhají zjednodušit složitost hledání. V případě rozvržení letadel na stojánky může heuristika zahrnovat například prioritizaci letadel, která nemohou stát na volné ploše, nebo upřednostnění letadel s kratším časem pobytu na stojánce. Další heuristikou může být snaha o přiřazení letadel k nejbližším volným stojánkům. Heuristické algoritmy jsou rychlé a flexibilní, protože se zaměřují na praktická pravidla a intuitivní přístupy, které fungují dobře ve většině případů. Avšak výsledky nejsou zaručeně optimální, protože heuristika obvykle ignoruje určité kombinace nebo scénáře, které by mohly vést k lepšímu řešení. Tento přístup je vhodný pro situace, kde je čas klíčový a dokonalá přesnost není nezbytná.

V neposlední řadě je možné použít určité formy **backtrackingu**. **Backtrackingu** je systematická metoda prozkoumávání všech možných řešení problému. V kontextu rozvržení letadel na stojánky algoritmus postupně zkouší přiřadit každé letadlo k dostupné stojánce, a pokud narazí na konflikt (například kolizi v čase nebo neplatnou stojánku), vrátí se zpět a zkouší jinou možnost. Tento přístup zaručuje nalezení optimálního řešení, pokud existuje, protože prozkoumá všechny možné konfigurace přiřazení. Jeho výpočetní složitost je však exponenciální, protože počet možných přiřazení roste rychle s počtem letadel a stojánek. **Backtracking** je proto vhodný pro malé až středně velké problémy nebo pro situace, kdy je potřeba absolutní přesnost. Pro větší instance problému je často kombinován s heuristikami, které omezují počet prozkoumávaných cest a zrychlují řešení.

3.2 Zvolený postup řešení

Při této práci se nejvíce osvědčil postup, který kombinuje **greedy** přístup s využitím určité formy heuristiky. Konkrétně se jedná o jakousi formu prioritizace letů letadel, které nemohou stát na volné ploše. Algoritmický by se tento postup mohl popsat následující posloupností činností:

1. Vyberu následující let z fronty letového plánu
2. Pokusím se ho přiřadit k jednomu z možných stojánek, daných typem letadla
3. Pokud se povedlu jdu zpět na 1.
4. Pokud může letadlo stát na volné ploše, přiřadím ho tam a jdu na 1.
5. Vyměním let s letem, z obsazených stojánek, který může být přiřazen na volnou plochu
6. Let, který byl odebrán ze stojánky dám na začátek fronty letového plánu
7. Jdu na 1.

Takovýto postup kombinuje výhody, jak **greedy** tak heuristického přístupu. Je výpočetně velmi rychlý, ale nalezené řešení je pouze pseudoptimální. Úspěšnost řešení je velmi závislá na vstupních datech. Pro specificky zvolená vstupní data může například dojít k tomu, že některé lety budou muset být zrušeny. Tento případ nastane ve chvíli, když letadlo, které nemůže stát na volné ploše, se pokusí vyměnit s letadly na stojánekách, ale ta také nebudou moci stát na volné ploše. Přesto si myslím, že v kontextu této práce je toto řešení dostačující a pro standardní data poskytuje očekávané výsledky.

Při další práci na této aplikaci by bylo vhodné ještě více zlepšit heuristické vlastnosti tohoto přístupu. Jedna z možností, co by se nabízela pro využití, je obsazování stojánek podle délky letu. Tento přístup by prioritizoval krátké lety na stojánekách a zamezoval by dlouhodobému obsazení stojánek. Ve finální verzi této aplikace je využito seřazení zadaných letů podle celkového času, který stráví na letišti (**přílet** - **odlet**). Takto zvolené seřazení částečně simuluje výše zmíněnou heuristiku a je velmi vhodné pro náhodně generovaná data, kde časy příletu a odletu jsou náhodně generované a letadla mohou strávit na letišti libovolný čas. Při vyhodnocování obsazení stojánek není vhodné mít jednu stojánku obsazenou jen jedním letadlem, což by tento přístup mohl řešit.

4 Programová dokumentace

Program je podle logiky znalostních systémů rozdělen do šesti navzájem propojených modulů. V následujících sekcích jsou jednotlivé moduly detailně popsány.

4.1 InferenceMachine

Modul **InferenceMachine** představuje inferenční modul ve znalostním systému. Slouží jako centrální řídicí jednotka v expertním systému pro letiště, který spravuje tok dat, výpočty a vysvětlení související s plánováním letů a přidělováním bran. Integruje různé komponenty, jako je modul pro interakci s uživatelem, modul pro správu

dat o letišti a letech, modul pro řízení obsazenosti bran a správu letů, modul pro řešení přidělování bran a modul pro poskytování detailních vysvětlení rozhodnutí. Díky strukturovanému workflow modul `InferenceMachine` zajišťuje vstup, zpracování a výstup dat, což umožňuje efektivní a vysvětlitelný provoz systému plánování letů na letišti.

4.2 `UserInterface`

Tento modul představuje komunikační modul ve znalostním systému a definuje třídu `UserInterface`, která slouží jako komunikační rozhraní mezi uživatelem a programem řídícím systém znalostní báze. Třída zajišťuje úlohy spojené s výpisem dat na příkazovou řádku i do výstupního souboru. Další důležitou funkcí této třídy je získávání uživatelských dat, zejména z příkazové řádky. Modul obsahuje také funkce nezbytné pro čtení a zpracování souborů, pro úspěšné načtení vstupních souborů daných parametry programu.

4.3 `SolutionGenerator`

Tento modul představuje modul řešení specifického problému znalostního systému. Klíčovým úkolem je plánování letů na letišti, přičemž vyhodnocuje jednotlivé lety podle kapacity letiště a řeší případné konflikty. Ve finální části se provádí třídění a shrnutí výsledků plánování. Nakonec se seznam výsledných letů přiřazených k jednotlivým branám třídí podle času příletu, což usnadňuje jejich správu a zajišťuje, že jednotlivé lety jsou uspořádány chronologicky. Pro samotné nalezení výsledku využívá tento modul datové struktury a metody definované v bázi faktů.

4.4 `ExplanationModule`

Tento modul představuje vysvětlovací mechanismus a slouží k vytváření vysvětlení a podrobných informací o stavu letů, včetně jejich přiřazení ke stojánkům, důvodů zrušení či odbavení na volné ploše. Umožňuje vyhledávat lety podle jejich ID a poskytuje uživatelsky přívětivé vysvětlení o tom, proč byl let přiřazen ke konkrétnímu místu, přesunut na volnou plochu nebo zrušen. Modul využívá faktické informace o aktuálním obsazení stojánek a pravidlech pro přiřazení letadel. Obsahuje metody pro podrobné ověření konfliktů v obsazení stojánek, důvody nemožnosti jejich využití a další relevantní detaily. Výstupem jsou čitelné textové zprávy, které pomáhají pochopit rozhodnutí systému plánování letů.

4.5 `KnowledgeBase`

Tento modul představuje bázi znalostí letištního systému. Obsahuje všechny klíčové funkce, pomocí kterých systém provádí výpočty. Tyto funkce slouží ke vytváření obsazení jednotlivých stojánek plánovanými lety. Tyto funkce slouží k nalezení pseudoptimálního řešení obsazení stojánek, pomocí výše zmíněných algoritmů. Tyto funkce fungují v kombinaci s modulem `SolutionGenerator`, který je zvenku využívá. Tyto funkce mohou být využity i pro poskytování vysvětlení.

4.6 `FactsBase`

Modul slouží jako báze faktů letištního systému. Obsahuje všechna data, která uživatel poskytl systému buď souborovým vstupem, nebo příkazovou řádkou. Modul zapouzdřuje tato data a poskytuje je systému, aby s nimi mohl dále pracovat. Modul také obsahuje mechanismus pro generování náhodných vstupních dat, pokud dostane tento pokyn od uživatele.

4.7 `DataStructures`

Tato složka obsahuje čtyři třídy, které definují jednotlivé objekty, které se využívají v letištním znalostním systému. Jmenovitě se jedná o třídy, které popisují letadla, letiště, stojánky a jednotlivé lety.

5 Uživatelská dokumentace

Pro spuštění aplikace je nutné mít nainstalovanou Python verzi 3.12 a vyšší. Program nevyužívá žádné externí knihovny, které by nebyly součástí standardních knihoven této verze Pythonu. Program se spustí z příkazové řádky příkazem:

python main.py (inicializační soubor) (soubor s letovým plánem)

Aby tento příkaz fungoval, musí uživatel být v kořenovém adresáři projektu, kde se nachází soubor `main.py`. Program využívá nejvýše dva parametry. První z nich je povinný a definuje inicializační soubor, který obsahuje data, definující letiště a jednotlivá letadla (viz kapitola 1). Bez tohoto parametru program vypíše chybové hlášení a vypne se. Druhý parametr je volitelný a uživatel jej nemusí zadat, avšak pokud tento parametr zůstane prázdný, nebude možné načíst letový plán ze souboru. I přesto má uživatel možnost provést generování dat letového plánu a aplikace bude dále pracovat s těmito daty. Jakmile program má obě tyto datové sady (buď zadané parametrem nebo generované) může začít s výpočtem optimálního obsazení stojánek. Uživatel nemůže žádným způsobem ovlivnit samotný výpočet. Po jeho provedení program vypíše všechna načtená a napočtená data. Poté má uživatel možnost požádat systém o vysvětlení toho, jak došlo k výpočtu a zařazení jednotlivých letů. Program poskytuje uživateli nápovědu, jak se systémem pracovat a jaké vstupy jsou od něj očekávány. Program se ukončí zadáním textu "koniec".

Samotný běh systému se z pohledu uživatele dá rozdělit do dvou částí:

```
----- Vítejte v aplikaci plánovače odbavení letů na letišti! -----
---- Tato aplikace Vám poskytne optimální rozložení obsazení stojánek a volné plochy na letišti ----
-----

Ovládání:
Uživatel má možnost ovlivnit pouze s jakými daty program pracuje
Program provede výpočet obsazení stojánek sám
Zároveň program pracuje s množinou dat popisujících letadla a letiště ze souboru daného prvním argumentem programu
Tato data lze změnit v souboru config/init.txt
Uživatel má možnost rozhodnout zda-li program využije úkázková data ze souboru config/table.txt
Nebo lze vygenerovat svoje vlastní náhodná data
Po provedení výpočtu může uživatel požádat o vysvětlení daného letu zadáním jeho indexu
Všechny výpisy jsou zároveň prováděny do souboru výsledek.txt a po ukončení programu jsou zde k nalezení

Přejete si vygenerovat data pro lety nebo využít základní data? (1 - generovat, 2 - soubor): 1
Proběhne generování! Zadejte počet letů, které chcete vygenerovat (celé číslo 1-100): 50
```

Obrázek 2: Spuštění systému.

```
Zadejte číslo letu, pro který chcete dostat vysvětlení (celé číslo) nebo ukončete program ("koniec"): 16

Let: [16] Cessna Citation X, Přílet: 00:10, Odlet: 17:13
Let s daným indexem byl odbaven na volné ploše!
Let nemohl použít stojánek: Stojánek: s1, protože byl obsazen lety: Let: [2] Embraer E190, Přílet: 07:04, Odlet: 12:35.
Let nemohl použít stojánek: Stojánek: s2, protože byl obsazen lety: Let: [18] Bombardier CRJ700, Přílet: 02:25, Odlet: 05:30
Let nemohl použít stojánek: Stojánek: s3, protože byl obsazen lety: Let: [8] Airbus A350, Přílet: 04:41, Odlet: 04:56, Let:
Let nemohl použít stojánek: Stojánek: s4, protože byl obsazen lety: Let: [3] Boeing 737, Přílet: 06:22, Odlet: 11:08.
Let nemohl použít stojánek: Stojánek: s5, protože byl obsazen lety: Let: [9] Concorde, Přílet: 01:18, Odlet: 06:57, Let: [1]
Let nemohl použít stojánek: Stojánek: s6, protože byl obsazen lety: Let: [0] Gulfstream G650, Přílet: 02:31, Odlet: 13:26.
```

Obrázek 3: Dotazování systému.

Po spuštění systém vypíše informace o sobě a poskytne uživateli možnost výběru, s jakými daty poběží. Uživatel může zvolit generování dat nebo načtení dat z výše zmíněného souboru daného druhým parametrem. Pokud uživatel zvolí generování, systém se ho ještě dotáže, pro kolik záznamů má vytvořit náhodný letový plán (viz Obrázek 2). Druhá část programu, kdy je očekávána interakce s uživatelem, nastane po vypočtení optimálního obsazení stojánek. V tuto chvíli se může zeptat, proč systém rozhodl daným způsobem pro libovolné letadlo. Systém mu poté vypíše informace o daném letu a může probíhat další dotazování (viz Obrázek 3).

6 Zhodnocení výsledků

Výsledky programu jsou silně závislé na povaze vstupních dat. Speciálně v případě generování náhodných dat pro vyšší počty letů, není ani pseudoptimální řešení dostatečně uspokojivé. Program nalezne dobré řešení pro smysluplná vstupní data a to zpravidla ta náhodně vygenerovaná nejsou. Níže v tabulce jsou zaneseny výsledky, které popisují situaci na letišti s patnácti stojánky:

Počet letů	Počet letů na stojánkách	Počet letů na volné ploše	Počet zrušených letů
10	10	0	0
25	24	1	0
40	31	8	1
60	38	20	2
80	46	20	14
100	49	24	27
102 (ručně zadaná data)	84	17	1

Tabulka 1: Výsledky

Z vypočtených dat je jednoznačně vidět, že kapacita stojánek se velmi rychle vyčerpá a zároveň se často mohou vygenerovat letadla, která nemohou stát na volné ploše a tudíž se jejich let musí zrušit. V poslední řádce jsou pro srovnání uvedeny statistiky pro letový plán zadaný souborem, kde jednotlivé lety jsou rovnoměrně rozděleny mezi všechna letadla a rozdíly mezi příletem a odletem jsou poměrně malé. Z těchto měření vyšel fakt, že pro letové plány s více jak sto náhodně generovanými lety je aplikace pro patnáct stojánek už nepříliš použitelná. Tento fakt by šel změnit, pokud by se například upravil inicializační soubor a byly by přidány stojánky a k nim odpovídající letadla.

7 Závěr

Myslím si, že se mi dobře podařilo naplnit zadání práce, tedy vytvořit funkční simulaci znalostního systému pro správu letiště. Podařilo se mi vyřešit klíčové úlohy, jako především dekompozici programu do jednotlivých modulů znalostního systému, a také nalezení samotného efektivního algoritmu pro nalezení optimálního řešení. Při práci jsem nenarazil na žádný podstatný problém a celková tvorba systému byla velmi intuitivní. V aktuální chvíli se tomuto systému neplánuji dále věnovat, ale případné zlepšení by bylo možné provést v reprezentaci výsledků, respektive vytvoření specifického GUI.