VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SIMULACE SPOLUPRÁCE AGENTŮ V PROSTŘEDÍ JASON

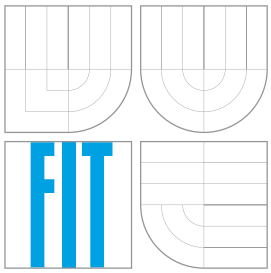
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB KŘÍŽ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SIMULACE SPOLUPRÁCE AGENTŮ V PROSTŘEDÍ JASON

SIMULATION OF COOPERATION OF AGENTS IN JASON ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB KŘÍŽ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KRÁL
SUPERVISOR

BRNO 2013

Abstrakt

Práce se zabývá tvorbou simulátoru multiagentního systému, ve kterém agenti spolupracují. Čtenář se nejprve seznámí se základy agentních systémů, jejich tvorbou a modelováním. V práci je popsán návrh prostředí, v němž se budou agenti pohybovat, a úlohy, jakou budou řešit. Dále je navrženo a implementováno chování tří režimů inteligence racionálních BDI agentů, s různou úrovní spolupráce — bez spolupráce, mírná spolupráce a komplexní spolupráce. Celý systém je realizován ve frameworku Jason a rozsáhle otestován. Na základě simulačních experimentů je vyhodnoceno chování a porovnána účinnost jednotlivých režimů inteligence. Dosažené výsledky dokazují převahu týmu, ve kterém agenti více spolupracují.

Abstract

This work deals with creation of simulator of multi-agent system in which agents cooperate. Reader is introduced to basics of agent systems, their creating and modeling. Design of environment in which agents exist and task they solve is described in this work. There are designed and implemented three levels of racional BDI inteligence of agents with different level of cooperation — without cooperation, weak cooperation and complex cooperation. Whole system is implemented in Jason framework and extensively tested. Behavior of agents is analyzed and levels of intelligence are compared based on simulation experiments. Achieved results prove dominance of team in which agents cooperate more.

Klíčová slova

multiagentní systém, umělá inteligence, Jason, AgentSpeak, simulace, spolupráce, BDI agent

Keywords

multi-agent system, artificial intelligence, Jason, AgentSpeak, simulation, cooperation, BDI agent

Citace

Jakub Kříž: Simulace spolupráce agentů v prostředí Jason, bakalářská práce, Brno, FIT VUT v Brně, 2013

Simulace spolupráce agentů v prostředí Jason

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Krále.

. . . . . .
Jakub Kříž
14. května 2013

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce panu Ing. Jiřímu Králi za cenné rady a připomínky při tvorbě této práce a za to, že mi umožnil využít školní stroj pro vykonaní experimentů.

Obsah

Kapitola 1

Úvod

Odjakživa se lidé snaží usnadnit si svou práci a zpříjemnit si tak život. Využívají k tomu nejrůznější pomůcky, od jednoduchých ručních nástrojů přes počítačové programy až po velice složité soustavy strojů. Lidé tyto přístroje neustále zdokonalují a snaží se, aby vykonaly co nejvíce jejich práce. Tato tendence vedla ke tvorbě samostatných systémů, tedy takových, které jsou schopny plnit požadované činnosti bez zásahu člověka. Jedním z těchto autonomních systémů jsou *agenti*. Agenti obvykle vykazují jisté známky inteligence a vykonávají určitou činnost v zájmu jejich vlastníka.

Agenti se v prostředí často sdružují do skupin, ve kterých ne vždy musí spolupracovat. Takové systémy se nazývají multiagentní. *Multiagentní systémy* jsou relativně novým oborem v oblasti umělé inteligence. Tento obor vznikl s rozvojem počítačových sítí a později internetu v 80. letech 20. století, kdy byly agenti využíváni pro distribuované výpočty. Dnes již tento přístup pronikl i do mnoha dalších odvětví, v nichž se využívá jeho decentralizované anebo paralelní činnosti.

Při tvorbě multiagentních systémů se nejprve vytvoří jejich počítačový model, který je umístěn do virtuálního prostředí, kde může být jednoduše simulován a otestován. Na základě těchto experimentů se analyzuje činnost systému, což umožní jeho další vývoj a úpravu ještě před nasazením do reálného světa a prostředí. V rámci této práce je právě takový simulátor multiagentního systému navržen a implementován, včetně kompletního prostředí a několika režimů činnosti agentů, a následně je otestován řadou experimentů.

Na začátku této práce v kapitole 2 jsou vysvětleny základní pojmy z oblasti multiagentních systémů, jejich členění a některé využívané principy. Kapitola 3 se zaměřuje na nejčastější typ racionálního agenta — BDI[1] agenta — a jeho realizaci v jazyce *AgentSpeak(L)*. Kapitola 4 se zabývá tvorbou modelů a jejich simulací.

Návrh prostředí, v němž se budou agenti pohybovat, a úlohy, kterou budou agenti řešit, je popsán v kapitole 5. V kapitole 6 jsou navrženy tři režimy činnosti agentů a stručně přiblíženo jejich chování. Režimy umělé inteligence se liší vzájemnou úrovní spolupráce mezi agenty v týmu. Jednotlivé režimy jsou: bez spolupráce, mírná spolupráce a komplexní spolupráce. Tvorba celého simulátoru a implementace navrženého multiagentního systému ve frameworku Jason je uvedena v kapitole 7. V kapitole 8 jsou popsány experimenty a je provedeno srovnání a analýza jednotlivých režimů. V závěru práce jsou shrnuty dosažené výsledky a zmíněny možnosti dalšího vývoje a využití systému.

[1] Belief - Desire - Intention – architektura agenta založená na představách, touhách a záměrech

Kapitola 2

Agentní systémy

Tato kapitola se zabývá úvodem do agentních a multiagentních systémů. Bude zde vysvětlen pojem inteligentní agent a popsány jeho klíčové vlastnosti. Dále bude představeno několik typů agentů a jejich základní principy. Poté následuje dělení prostředí, ve kterém se agent může nacházet. Závěrem bude shrnut agentní systém jako celek.

2.1 Inteligentní agent

Slovo agent vychází z latinského *agentum*, což znamená „ten, kdo jedná“. Obecně je agent chápán jako někdo, kdo v určitém prostředí jedná v zájmu svého klienta.

Pojem agent v oblasti informačních technologií obvykle označuje umělého agenta. Umělý agent je systém (softwarový nebo hardwarový) vytvořený člověkem k nějakému předem zamýšlenému účelu. V prostředí, v němž je umístěn, jedná agent samostatně ve prospěch svého klienta a řeší zadanou úlohu bez zásahu člověka, včetně podúloh a konfliktů, které nastanou. Schopnost samostatného jednání je nejdůležitější vlastností agenta, jež ho odlišuje od jiných pasivních prvků.

Jako inteligentní je považován agent, který vykazuje inteligentní chování. Protože je tato definice příliš vágní, musí mít inteligentní agent dle Wooldridge a Jenningse [7] tyto vlastnosti:

- **Autonomie** – schopnost nezávisle jednat, mít plnou kontrolu nad svým jednáním
- **Reaktivita** – schopnost pohotově reagovat na změny prostředí
- **Proaktivita** – schopnost přebírat iniciativu, svým jednáním ovlivňovat prostředí v souladu se svými cíli
- **Sociálnost** – schopnost interagovat s ostatními agenty

2.2 Typy agentů

Existuje více typů agentů, kteří se liší vnitřní architekturou, prací se znalostmi a mechanismy rozhodování. U některých agentů jsou požadované vlastnosti vhodně nakombinovány a vznikají tak hybridní agenti.

Můžeme rozlišit několik základních typů agentů:

- **Reaktivní agent**

  Tento typ agenta má nejjednodušší vnitřní architekturu. Neobsahuje plánovací mechanismus ani žádnou vnitřní reprezentaci prostředí, pouze bezprostředně reaguje na jeho změny tak, aby dosáhl cíle. Jeho akce nejsou výsledkem výpočtů či dedukcí na základě znalostí, ale pouze reakcemi na podněty v závislosti na jeho vnitřním stavu. Čistě reaktivním agentem je takový, který nemá vnitřní stav a rozhoduje pouze na základě podnětů.

- **Deliberativní agent**

  Agent uvažující o svých dlouhodobých záměrech. Plánuje postup svých akcí za účelem dosažení cíle. Udržuje si vnitřní reprezentaci prostředí, vnitřních stavů a znalostí, na základě kterých se rozhoduje. Pro zvolení nejvhodnější akce a dalších plánů musí být agent schopen různých výpočtů, které představují jeho vnitřní činnosti. K dosažení svých záměrů pak agent ovlivňuje okolní prostředí tak, aby získal nějakou výhodu.

- **Kognitivní agent**

  Agent, který má schopnost vyvozovat logické závěry ze svých pozorování okolního prostředí. Musí být především schopný se učit a uchovávat informace. Během svého působení si vytváří svou vlastní databázi znalostí, do které si ukládá informace získané z prostředí a vlastní logické dedukce.

- **Racionální agent**

  Agent s praktickým uvažováním. Je považován za nejsložitější a nejdůmyslnější typ agenta, který stojí nejvýše v pomyslné hierarchii agentů. Kombinuje v sobě výhodné vlastností předchozích typů agentů. Jeho struktura obsahuje jak plánovací jednotku, tak i kognitivní jednotku včetně báze znalostí. Tento agent je schopen se učit na základě svých poznatků a pak plánovat svoji činnost, aby dosáhl svých cílů racionálním způsobem. Racionálním způsobem je myšleno takové jednání, které v daný okamžik a s danými znalostmi povede nejlépe k dosažení cíle.

V rámci této práce jsou vytvořeni BDI agenti (viz kapitola 3), kteří se řadí mezi agenty racionální.

2.3 Prostředí

Agentní systém je tvořen agentem a prostředím. Prostředí je veškerý okolní svět, se kterým přichází agent do styku během své činnosti. Prostředí má své vnitřní stavy, kterými je reprezentováno. Pánové Russell a Norvig [5] dělí prostředí podle jeho povahy následovně:

- **Diskrétní x spojité**

  Diskrétní prostředí má konečně nebo spočetně mnoho stavů. Na číslicových počítačích je každé prostředí diskrétní.

- **Plně x částečně pozorovatelné**

  V plně pozorovatelném prostředí může agent kdykoliv sledovat jeho kompletní stav.

- **Statické x dynamické**

  Statické prostředí je takové, které se mění pouze akcemi agenta.

- **Deterministické x nedeterministické**

  V deterministickém prostředí má každá akce jeden předpokládaný výsledek. Následný stav prostředí je predikovatelný z momentálního stavu prostředí a akce vykonané agentem.

- **Epizodní x neepizodní**

  V epizodním prostředí je možné běh systému rozdělit na sobě nezávislé části - epizody. Po ukončení jedné epizody lze začít novou od počátečního stavu.

- **Strategické**

  Prostředí, které spolu sdílí více agentů. Statické a deterministické prostředí se může agentovi jevit jako dynamické a nedeterministické, protože jej mohou svou činností ovlivnit ostatní agenti.

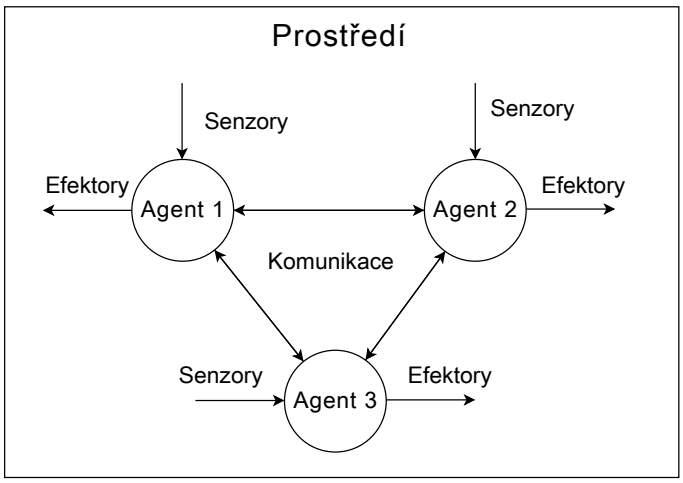
Prostředí navržené v kapitole 5 je diskrétní, částečně pozorovatelné a neepizodní. Protože se v něm pohybuje více agentů jedná se o prostředí strategické, jinak by bylo statické a deterministické.

2.4 Agentní a multiagentní systém

Obecný systém je definován jako souhrn souvisejících prvků, které jsou sdruženy do smysluplného celku.

Agentní systém je podle Wooldrige [8] chápán jako systém, který obsahuje agenta a prostředí. Všechny prvky (aktivní - agenti, pasivní - neagentní) tohoto systému společně vytvářejí prostředí, ve kterém se agent nachází. Pro interakci s prostředím slouží agentovi senzory a efektory. Senzory vnímá stav prostředí, efektory používá agent k vykonání požadované akce, která má vést k ovlivnění prostředí. Po provedení akce přejde prostředí, na základě původního stavu a vykonané akce, do stavu jiného.

Jestliže je v agentním systému více než jeden agent, jedná se o multiagentní systém [6]. Kromě ovlivňování prostředí jsou agenti také schopni vzájemné interakce, mohou spolu komunikovat a spolupracovat (viz obr. 2.1).



Obrázek 2.1: Multiagentní systém.

Kapitola 3

BDI agent a AgentSpeak(L)

Tato kapitola se zaměřuje na BDI agenta. Jedná se o dnes nejrozšířenější přístup k realizaci racionálního agenta, protože poskytuje dostatečnou abstrakci chování agenta. První část kapitoly bude věnována BDI architektuře, na které je tento agent postaven. V druhé části kapitoly bude popsána jeho realizace v systému *AgentSpeak(L)* a implementace v jazyce *Jason*.

3.1 BDI architektura

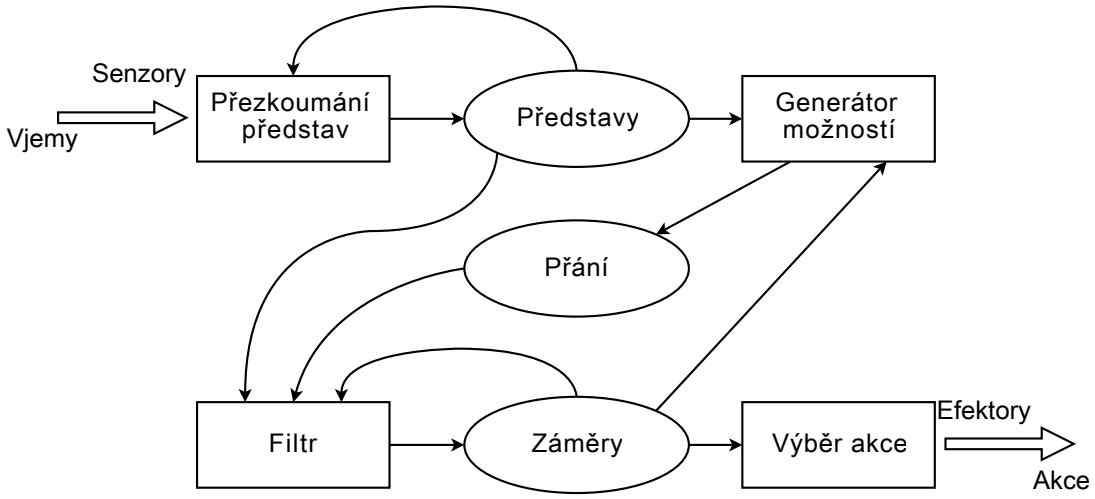
Tato architektura se snaží vycházet z principů podobných lidskému uvažování. Každý člověk se při svém rozhodování a uvažování řídí svými mentálními postoji. Jsou to převážně představy, přání a záměry. V agentních systémech jsou dle [2] definovány následovně:

- **Beliefs** – představy
  Představy jsou agentovy informace o stavu prostředí, ve kterém se nachází. Agent je může získat pomocí senzorů z prostředí, sám vydedukovat na základě jiných znalostí nebo obdržet od jiného agenta. Nemusejí být nutně pravdivé ani trvanlivé, jedná se pouze o subjektivní poznatky.

- **Desires** – přání
  Přání neboli touhy představují všechno, čeho by chtěl agent svou činností dosáhnout. Jsou to cíle agenta, které se snaží následovat a pokud možno splnit. Agentovi touhy mohou být i protichůdné.

- **Intentions** – záměry
  Záměry přestavují aktuálně zvolená přání jako cíle, které hodlá agent splnit. Jedná se vlastně o agentovy aktuální plány. Splnění záměru vede k přiblížení nebo naplnění touhy agenta. Záměr přetrvává, dokud není splněn nebo pokud jej nelze splnit.

Na základě těchto mentálních postojů se agent rozhoduje a řídí své chování. Na obrázku 3.1 je zobrazen mechanismus rozhodovacího procesu BDI agenta. Tento proces je opakován ve smyčce, dokud agent nedosáhne všech svých přání. Nejdříve agent získá ze svých senzorů vjemy z prostředí, podle kterých aktualizuje své představy. Na základě nových vjemů přehodnotí své stávající přání a záměry. Přání jsou vytvářena podle představ a záměru, které agent má. Mohou vzniknout nové záměry na základě představ, přání a aktuálních záměrů nebo mohou být záměry označeny za nesplnitelné a odstraněny. Poté jsou vybrány



Obrázek 3.1: Rozhodovací proces BDI agenta.

nejvhodnější záměry a pomocí efektorů provedena zvolená akce, aby těchto záměrů mohlo být dosaženo.

3.2 AgentSpeak(L)

AgentSpeak(L) je formální systém založen na BDI architektuře, který je určen pro realizaci BDI agenta. Jedná se o konkrétnější popis BDI architektury, který popisuje zejména sémantiku jejich funkcí a přibližuje syntaxi jazyka pro tvorbu BDI agentů. Představuje tak teoretický popis implementace BDI agenta, ale na dostatečně vysoké úrovní abstrakce, na níž zachycuje pouze základní rysy. Jeho syntaxe vychází z logického programovacího jazyka *Prolog*. K reprezentaci postojů (představ, přání a záměrů) používá modálních operátorů. Dále jsou popsány základní složky tohoto jazyka dle literatury [1] a [4].

Představy a pravidla

Všechny představy jsou ukládá do báze znalostí jako formule predikátové logiky. Atomická představa nebo její negace jsou literály, které se mohou skládat a tvořit tak složitější představy. Představy v bázi znalostí považuje agent za pravdivé, všechny ostatní za nepravdivé. Při dotazu na bázi znalostí se používá unifikace s prvním správným, kdy se najdou vhodné představy, které nahradí neznámé hodnoty tak, aby tvrzení platilo.

Pravidla slouží pro dedukování dalších informací na základě představ, podle daných zákonitostí a závislostí. Zjednodušují práci s bází znalostí a usnadňují získávání potřebných dat z více různých navzájem závislých představ.

Cíle

Cíle označují stavy, kterých chce agent dosáhnout. Existují dva druhy cílů. Prvním druhem je cíl testovací, který slouží ke zjištění informací z báze znalostí. Druhým typem je cíl dosažení, u kterého se agent bude snažit dosáhnout změny prostředí, jež by odpovídala cíli.

Plány a události

V jazyce AgentSpeak(L) je plán neboli záměr postup činností, které musí agent vykonat pro dosažení cíle. Tento plán budoucích činností slouží k udržení kontextu vykonávané akce a k dosažení dlouhodobých přání. Plán se skládá ze tří složek:

- **Spouštěcí událost**
  Spouštěcí událost je akcí, která vede k vykonání plánu. Tato akce je generována jako reakce na změnu cílů nebo vybraných přestav. Název spouštěcí události odpovídá názvu představy, na jejíž změnu reaguje, nebo názvu cíle, kterého se dosáhne uskutečněním plánu.

- **Kontext**
  Jedna spouštěcí událost může generovat více různých možností — relevantních plánů. Z těchto možností se vybírá první aplikovatelný plán. Pro zjištění aplikovatelnosti plánu, který se má vykonat, slouží kontext. Kontext přestavuje podmínku tvořenou konjunkcí literálů a vztahů mezi nimi. Aby byl plán aplikovatelný musí být podmínka v kontextu vyhodnocena vzhledem k aktuálnímu stavu báze znalostí jako pravda.

- **Tělo plánu**
  Tělo plánu tvoří posloupnost nových cílů nebo akcí, jejichž provedením dosáhne agent požadovaného cíle. Přidáním dalších cílů v těle plánů se spustí vykonání dílčích plánů, jež společně vedou k úspěšnému dosažení prvotního cíle. Jak se plány do sebe noří, vytváří se postupně zásobníková struktura cílů, kterých je třeba ještě dosáhnout. Nemožnost splnění jakékoli části těla plánu znamená neúspěšnost splnění celého plánu a nedosažitelnost jeho cíle, který je následně odebrán. Na to musí agent vhodně zareagovat a přejít na plán jiný.

3.3 Jason

Jason je framework pro realizaci multiagentního systému, který je vytvořen v jazyce Java. Jako jazyk pro programování agentů používá rozšířenou realizaci teoretického jazyka AgentSpeak(L). Hlavním rozšířením jsou neblokující plány, které umožní agentům paralelně plnit více plánů. Dalším rozšířením je přidávání nepovinných anotací k představám nebo plánům. Anotace slouží jako poznámky a doplňující informace, jež zpřehledňují a ulehčují tvorbu agenta, ale nijak nezvyšují vyjadřovací schopnost jazyka.

Systém Jason také dovoluje podrobně specifikovat model agentova prostředí. Programátor si může podle svých potřeb vytvořit kompletní prostředí a akce, které ho ovlivňuji. Veškerá implementace prostředí se zhotovuje jednoduše v jazyce Java. Systém Jason potom zprostředkovává vzájemnou komunikaci mezi modulem inteligence a modulem prostředí. Deleguje vybrané akce agenta na akce v prostředí a předává agentovy zvolené vjemy z prostředí.

Kapitola 4

Modelování a simulace

Tato kapitola věnuje teoretickému základu z oblasti modelování a simulace. Simulace má rozsáhlé využití téměř ve všech odvětvích, v nichž je potřeba analyzovat chování nějakého systému za pomoci jeho modelu. Na rozdíl od experimentování s originálním systémem je práce s modelem levnější, rychlejší a často jediný způsob získání informací. Vhodná je také pro usnadnění zkoumání složitých systémů nebo při vývoji nových zařízení, kdy jednoduchá úprava modelu je maličkostí proti změně modelovaného fyzického systému. Nevýhodou tohoto přístupu je zejména často náročné nebo nemožné dokázání platnosti modelu nebo vysoké požadavky na výpočetní výkon simulace.

Na začátku kapitoly bude vysvětlen pojem model včetně jeho vlastností a klasifikace. Dále budou popsány principy modelování a simulace. Poslední část kapitoly se bude zabývat důležitostí simulace v agentních systémech.

4.1 Model

Model je napodobenina systému jiným systémem. Model je proti vzoru zjednodušený. Zachovává si pouze ty vlastnosti, jež jsou považovány za podstatné a důležité, ostatní jsou zanedbány nebo velmi zjednodušeny. Díky tomu se s modelem snáze pracuje, ale vždy se jen přiblíží k originálu, nikdy není shodný. Kvalita modelu je závislá na dostupných informacích o systému, který modelujeme, především pak na výběru jeho podstatných vlastností. Je snaha mít co nejkvalitnější model, protože chybný model dává při simulaci chybné výsledky.

Nejdůležitější vlastností modelu je jeho validita neboli platnost, tedy ověření zda model odpovídá systému nebo se v požadovaných situacích chová stejně. Často lze validitu dokázat jen velmi těžko nebo vůbec, proto se provádí částečné dokazování. Částečné prokazování validity spočívá v tom, že se porovnávají známé informace o původním systému s výsledky jednoduchých experimentů na modelu. U validního modelu musí být hodnoty stejné nebo v povolené toleranci.

Tradičně dělíme modely na spojité, diskrétní a kombinované. Spojité jsou modely, které svůj stav mění spojitě. Jsou popsány například diferenciálními rovnicemi. U diskrétních modelů se naopak proměnné modelu mění skokově. Modely kombinované obsahují spojité i diskrétní prvky současně v jednom modelu.

4.2 Modelování

Modelování je proces vytváření modelu. Tento proces se skládá z několika technik, jejichž cílem je vytvořit abstrakci reality vhodnou pro simulaci. Modelování je obecně dosti náročná činnost, která pro vytvoření kvalitního modelu často vyžaduje spojit znalosti z různých oblastí. Nejobtížnější problém modelování je validita modelu, protože nelze absolutně prokázat přesnost modelu.

Prvním krokem modelování je důkladná analýza modelovaného systému. Shromáždí se veškeré znalosti o systému, z nichž se vyberou klíčové složky a důležité vlastnosti. Dále se musí ujasnit, co se od výsledného modelu očekává a k čemu bude sloužit. Na základě této analýzy se vytvoří zjednodušený *abstraktní model*, jenž slouží jako předloha pro *simulační model*. Simulační model je shodný s abstraktním, jde jen o přenos abstraktního modelu do simulačního prostředí. Simulační model je obvykle spustitelný program, kterému můžeme zadávat parametry případně ho řídit. [3]

4.3 Simulace

Simulace je metoda získávání nových znalostí o systému experimentováním s jeho modelem. Prostřednictvím validního modelu můžeme nepřímo zkoumat původní systém. Pro účely simulace musí být model popsán odpovídajícím způsobem. Ne každý model je vhodný pro simulaci.

V rámci simulace se opakují simulační experimenty s různými parametry. Řeší se simulační model a sleduje se vliv parametrů na jeho chování. V průběhu simulace se sbírají důležitá data a jejich analýza poskytuje nové znalosti o systému. Simulace také slouží k ověřování hypotéz a pomáhá navrhovat úpravy modelu, případně původního systému, vedoucí k nějakému zlepšení.

Mezi simulacemi se neustále provádí konfrontace získaných údajů se známými fakty o modelovaném systému, čímž se ověřuje validita modelu.

4.4 Simulace v agentních systémech

V oblasti agentních systémů je simulace velice důležitá. Často se jedná o jediný způsob, kterým lze ověřit racionálnost rozhodování agenta v libovolných situacích a nejrůznějších stavech prostředí. Simulaci je obvykle možné libovolně přerušit a důkladně prostudovat aktuální rozhodovací proces agenta, včetně jeho mentálních stavů, a v závislosti na momentálním stavu prostředí vyhodnotit korektnost rozhodnutí, jež agent právě udělal. Během simulace je také možné uchovávat seznam změn stavů prostředí a akcí agenta, který se potom může zpětně analyzovat a zhodnotit zda jsou rozhodnutí agenta racionální i v dlouhodobějším kontextu. Velký význam má i tvorba statistik, kdy se během simulace poznamenávají podstatné údaje o jejím průběhu a činnosti agenta. Po skončení simulace se z těchto dílčích informací vytvoří statistický přehled, na jehož základě lze rychle porovnat více agentů nebo vyhodnotit celkové chování agenta během simulace.

Kapitola 5

Návrh úlohy agentů

Jedním z cílů mé bakalářské práce bylo navrhnout prostředí, ve kterém se budou agenti pohybovat, a úlohu, již bude tým agentů řešit. V úloze měl být kladen důraz na kooperaci agentů v týmu.

V úloze, kterou jsem navrhl, proti sobě soupeří dva týmy agentů. Jejich soupeření spočívá ve vzájemném ničení věží a jejich následném opravování. V následujících podkapitolách bude podrobně popsáno mnou navržené prostředí a úloha včetně jejích pravidel. Dále budou rozebrány všechny elementy, které se mohou v prostředí vyskytnout, jejich možnosti a akce.

5.1 Prostředí

Oba týmy agentů se pohybují po jedné 2D mapě. Mapa je horizontálně i vertikálně rozdělená na stejné části, čímž se vytvoří šachovnicová síť se stejně velikými políčky. Na mapě jsou dva typy terénu: louka a zeď. Každé políčko prostředí má po celou dobu simulace právě jeden z nich. Převládajícím typem terénu je louka. Na louce se odehrávají veškeré činnosti agentů a stojí na ní všechny budovy. Po těchto políčkách se mohou agenti libovolně pohybovat. Druhým typem terénu je zátaras neboli zeď. Jak už sám název napovídá, toto políčko je pro agenta nedostupné. Nemůže na něj nikdy vstoupit ani jej přeskočit. Jediným způsobem, jak se agent může dostat na políčko louky za zdí, je najít volnou cestu kolem zdi. Aby byla mapa pro moji úlohu použitelná, musejí být tyto dva typy terénu vhodně nakombinovány. Na mapě se nesmí vyskytnout žádná nedostupná část, takzvaný ostrůvek, což je oblast louky, která je dokola obehnána zdí. Celá mapa musí být navíc po obvodu ohraničena zdí, aby se agent nepokoušel jít mimo ni.

5.2 Objekty

Na mapě se může vyskytnout několik typů objektů, které se dají rozdělit na dvě skupiny: agenty a budovy. Budovy jsou objekty, které označují jedno políčko na mapě, kde může agent provést důležité akce. Vlastnictví a typ budovy ovlivňuje, jaké akce v ní agent může provést. Každá budova po celou dobu náleží jednomu týmu a nemění svou polohu. Máme dva typy budov: věž a základnu.

Po celou dobu simulace musí mít každý objekt, ať agent nebo budova, na mapě jasně danou pozici, která je vyjádřena souřadnicemi políčka, na němž objekt stojí. Na jednom

políčku může stát současně více agentů, může to být i políčko obsazeno budovou. Více budov však na jednom políčku stát nesmí.

Agent

Jediný objekt, který se může pohybovat. Veškerá jeho činnost je řízena umělou inteligencí (viz kap. 6). Vždy proti sobě soupeří dva týmy agentů, kteří jsou všichni v rámci jednoho týmu totožní. Po celou dobu zůstává počet agentů neměnný, stejně jako jejich týmová příslušnost. Agenti soupeřících týmů se nemohou navzájem přímo ovlivňovat. Mohou se ovlivňovat jen nepřímo skrze přítomnost v okolí budovy, více v kapitole 5.6. V každém kroku simulace musí každý agent vykonat jednu akci (viz kap. 5.4). Pouze tyto akce mohou ovlivnit prostředí. Agent získá po každé akci informace o prostředí (viz kap. 5.5), na základě kterých se rozhoduje o dalších akcích. Agent může nést maximálně jeden nástroj (viz kap. 5.3), tento nástroj může vzít nebo použít pouze na příslušných místech. Nástroj nelze mezi agenty předávat; bylo by to zbytečné, protože všichni mají stejnou rychlost a kapacitu.

Věž

Budova, o niž se soupeří. Na začátku soupeření mají oba týmy stejný počet věží, každá věž má maximální počet životů, který je pro všechny stejný. Agent může zaútočit na nepřátelskou věž, a tím ji ubrat vždy stejnou část životů (15 % maximální hodnoty). Když počet životů klesne na nulu, je věž zničená. Po jejím zničení se na její pozici objeví pět kusů suti, které může kdokoli sebrat. Tým, jenž zničí všechny soupeřovy věže jako první, vyhrává. Agent může svoji věž opravit, čímž ji přidá vždy stejnou část života, která je shodná s ubranou částí života při útoku. Nelze opravit zničené věže nebo přidávat život nad maximální hodnotu.

Základna

Domovská budova každého týmu, kterou nelze zničit. Má roli skladiště, kde mohou agenti uložit libovolný počet surovin a nástrojů. Na své základně mohou agenti také nástroje vytvářet, pracovat jich může více současně. Základna může být nepřátelským týmem vykradena.

5.3 Nástroje

Nástroj je věc, kterou může agent vzít, přenést a použít. Agent může nést pouze jeden nástroj. Vlastnictví nástroje mu rozšiřuje množinu možných akcí. Nástroje se skladují ve vlastní základně, nikde jinde nemohou být uloženy. Nástroj většinou slouží k přidání nebo odebrání života věži. Po použití nástroj mizí. Kromě obvyklých způsobů získání nástroje, které jsou popsány níže, může být nástroj ukraden ze soupeřovy základny. V úloze se používají tyto druhy nástrojů:

- **Zbraň**
  Nástroj určený k ničení. Jeho použitím může agent sníží počet životů vybrané cizí věže. Zbraň může agent vytvořit sám ve vlastní základně, nepotřebuje k tomu žádné suroviny jen čas. Zbraň způsobí věži vždy konstantní poškození.

- **Suť**
  Suť není obvyklý nástroj, je to surovina pro výrobu cihly. Vznikne jen na místě, kde stála věž. Po zničení věže se vytvoří pět kusů suti. Poté mohou agenti libovolného týmu přenést suť na svoji základnu.

- **Cihla**
  Tento nástroj slouží k opravení své poškozené věže. Cihlu může agent vytvořit na vlastní základně, k čemuž potřebuje suť. Agent může cihlu použít jen na své věži, čímž jí zvýší množství života o konstantní hodnotu.

5.4 Akce

Průběh celé úlohy je složen z jednotlivých kol. V každém kole musí každý agent vykonat nějakou akci, přičemž všechny trvají jedno kolo. Pomocí akcí mění agenti stav prostředí. Nikdo jiný akce provádět nemůže.

Agenti mohou vykonat tyto akce: vzít nebo uložit nástroj, zaútočit nebo opravit věž, pohnout se, vytěžit suť, vykrást základnu, vyrobit zbraň nebo cihlu a prázdná akce. Například když chce agent zaútočit na cizí věž, musí na ní stát a mít u sebe zbraň. Po provedení akce nemá agent žádný nástroj a je snížena hodnota života věže. Tato akce, stejně jako některé delší, je navíc podmíněna poměrem počtu agentů v okolí, více v kapitole 5.6. Popis všech akcí včetně podmínek je uveden v příloze C.

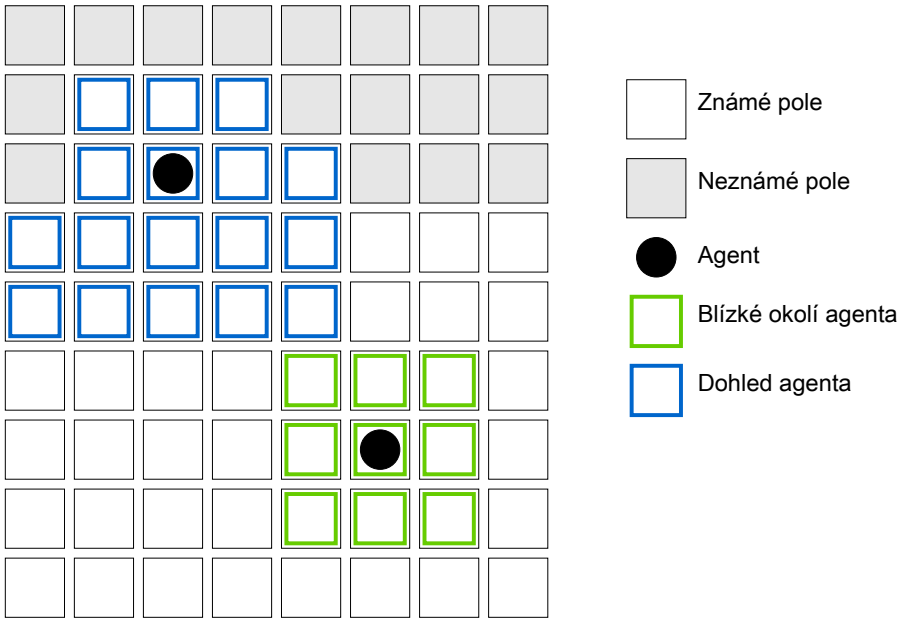
5.5 Vjemy z prostředí

V této úloze agent nikdy nedostává všechny informace o celém prostředí a jeho aktuálním stavu. Získává informace o prostředí buď přímo jen ze svého blízkého okolí, nebo nepřímo od ostatních agentů v jeho týmu (dle režimu inteligence). Informace, které může agent získat, můžeme rozdělit na dva typy: statické a dynamické.

Statické informace jsou v čase neměnné. Jakmile je agent získá, jsou platné až do konce úlohy. Jedná se o rozvržení mapy, tedy typ terénu jednotlivých políček a poloha budov. Tyto informace může přímo získat jen ze svého bezprostředního okolí, což je jedno políčko do všech osmi směrů od jeho aktuální pozice, viz blízké okolí agenta na obrázku 5.1. Na začátku agent nezná nic, ale jak postupuje prostředím objevuje a poznává políčka ve svém bezprostředním okolí. Po každém kroku se mu do jeho databáze znalostí přidá typ terénu a obsazení budovou na právě objevených políčkách.

Dynamické informace se na rozdíl od statických mohou změnit po každé akci. Jsou to zejména informace o aktuálním stavu jednotlivých budov, jako zdraví, počet nástrojů nebo zda je budova blokována. Tyto informace agent získává také ze svého okolí. Zde už se nejedná o bezprostřední okolí, ale o větší oblast tzv. dohled. Dohled (viz modré čtverečky na obrázku 5.1) je čtvercová oblast 5x5 políček kolem agenta, ale jsou to jen políčka, která agent už objevil a zná je, tedy ty, o nichž má statické informace. Po každém kroku tedy přidá nebo aktualizuje informace o stavu budov v jeho dohledu, a to včetně času, kdy tyto informace získal. Navíc po každém kroku simulace každý agent obdrží aktuální informace o stavu všech budov svého týmu, včetně jejich pozice.

Dalo by se říct, že se agent po každém kroku rozhlédne, pečlivě prozkoumá sousední políčka, čímž objeví budovy a pozná terén. Potom zjistí stav budov na objevených políčkách v jeho dohledu a dostane informace o stavu svých budov.



Obrázek 5.1: Okolí a dohled agenta.

5.6 Blokování budov

Všechny akce, které se provádějí v nějaké budově, mohou být zablokovány soupeřícími agenty, čímž si týmy mohou navzájem zamezit uskutečnění akce. O blokování budovy rozhoduje rozdíl počtu agentů jednotlivých týmů v okolí budovy. Za okolí budovy je považována celá oblast do vzdálenosti dvou políček od budovy, tedy čtverec 5x5 s budovou ve středu. Okolí budovy je tedy stejná oblast jako dohled agenta. To znamená, že je-li agent v okolí budovy, dohlédne na ni a může zjistit její stav, jenž sám svoji přítomností ovlivňuje. Následuje seznam všech podmínek pro provedení akce, které souvisí s blokováním budov. Agent může:

- opravit svoji věž, pouze pokud v jejím okolí není víc nepřátelských než přátelských agentů,
- zaútočit na cizí věž, pouze pokud v jejím okolí je více přátelských než nepřátelských agentů,
- udělat jakoukoli akci na své základně, pouze pokud v jejím okolí není o dva více nepřátelských než přátelských agentů,
- okrást cizí základnu, pouze pokud je v jejím okolí o dva více přátelských než nepřátelských agentů,
- vytěžit suť, pouze pokud je v jejím okolí více přátelských než nepřátelských agentů.

Kapitola 6

Návrh chování agentů

V této kapitole bude popsáno mnou navržené chování agentů, které by mělo vést k jejich vítězství. Vytvořil jsem tři režimy inteligence agentů s různou úrovní spolupráce — bez spolupráce, mírná spolupráce a komplexní spolupráce. Agenti ve všech těchto režimech mají některé prvky rozhodování stejné. Tyto principy budou popsány v první podkapitole. Poté následuje každému režimu věnovaná samostatná podkapitola, v níž bude popsán rozhodovací proces agenta s touto umělou inteligencí.

6.1 Společné principy

Celkové chování agentů z různých režimů je odlišné, přesto má některé principy společné. Jedná se o základní techniky obrany a hledání, které všechny režimy ve své činnosti běžně používají.

Měření vzdálenosti

Klíčové pro činnost agenta je určování vzdálenosti. Za vzdálenost se považuje délka cesty mezi políčky. Je-li známá cesta mezi počátečním a koncovým bodem, bere se její délka. Jinak se uvažuje čtyřnásobek přímé vzdálenosti, protože není známo rozložení terénu a předpokládá se následné zdržení při hledání cesty k cílovému políčku. Přímá vzdálenost je nejkratší teoretická cesta mezi políčky, tedy součet rozdílů jejich horizontálních a vertikálních souřadnic.

Obrana

Jako základní obranný mechanismus slouží zablokování budovy proti soupeřícímu týmu. Proto agent opustí okolí své budovy jen v případě, že to nezmění její stav vzhledem k blokování budov. To znamená, že agent svým odchodem z okolí budovy nesmí umožnit cizímu agentu, který je v okolí této budovy, provést na budově akci.

Zablokování

Když je pro agentovu akci zablokována jeho vlastní budova, zůstane na ní vyčkávat, dokud mu nebude akce umožněna. Nedělá nic pro to, aby odehnal soupeřící nebo přivolal své agenty. Aby tato situace nastala, musí být v okolí budovy víc nepřátelských agentů než vlastních, což dává jeho týmu výhodu v jiné části mapy. Tuto techniku nepřímé spolupráce využívají i agenti, kteří nespolupracují.

Hledání cesty

Agent skoro vždy zná pozici cíle, kam se má přesunout, a kde provést nějakou akci. Když zná agent cíl, pokusí se k němu najít nejkratší cestu přes objevená políčka. Jestliže cestu najde, vydá se po ni. V případě, že cestu nenalezne, znamená to, že mezi ním a jeho cílem existují nějaká neobjevená políčka. Tato políčka musí nějak systematicky prozkoumat, aby cestu našel. Dělá to tímto způsobem: Najde si objevené políčko louky, které je nejblíže cíle a sousedí s nějakým neobjeveným políčkem. Přesune se na toto políčko, čímž objeví alespoň jedno nové políčko. Tento postup opakuje, dokud neobjeví dostatek nových políček, po kterých by se mohl dostat k původnímu cílovému místu.

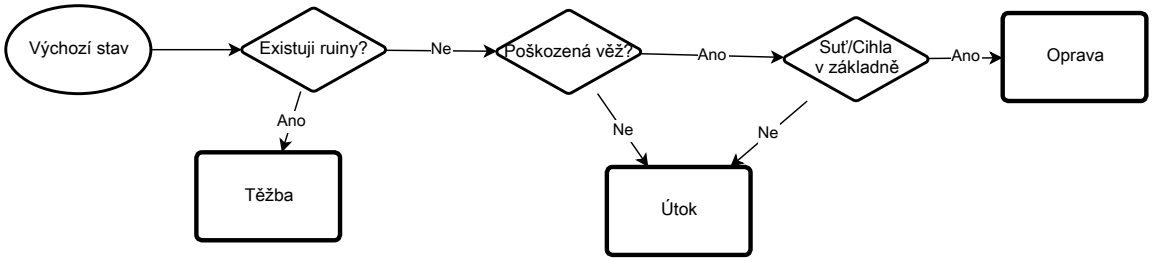
Vyhledávání cesty přes neobjevená políčka agent také praktikuje, je-li cesta přes objevená políčka dvakrát delší než přímá vzdálenost, ale jen když agent není blízko cíle. Je tedy dosti pravděpodobné, že existuje kratší cesta přes políčka neobjevená.

Jedinou výjimkou, kdy agent nezná konkrétní polohu cíle, je hledání nepřátelských věží nebo základny. Toto hledání agent provede podobně jako hledání cesty. Přesune se na sobě nejbližší políčko louky, které sousedí s neobjeveným políčkem. Tímto přesunem objeví a pozná minimálně jedno další políčko. Tuto činnost opakuje tak dlouho, dokud nenalezne to, co hledá, nepozná celou mapu nebo se nevyskytne něco důležitějšího na práci.

6.2 UI1 – bez spolupráce

Tento režim inteligence je nejjednodušší. Agenti vůbec neuvažují o ostatních agentech. Nijak spolu nekomunikují a nepředávají si tedy žádné informace o stavu prostředí. Chovají se, jako by v týmu byli sami.

Činnost agenta by se dala rozdělit do těchto tří dlouhodobějších plánů: těžba suti, oprava vlastní věže, útok na soupeřovu věž. V závislosti na prostředí si agent vybere jeden z těchto plánů a pokusí se ho splnit. Okamžitě po jeho splnění nebo selhání vybírá další, což opakuje až do konce úlohy.



Obrázek 6.1: Základní princip výběru dlouhodobého plánu agenta UI1.

Těžba

Nejprioritnějším plánem je těžba suti, protože se jedná o jediný omezený zdroj. V tomto režimu inteligence může jedině tento plán přerušit vykonávání ostatních plánů. Okamžitě po tom, kdy agent zjistí, že po nějaké zničené věži zůstala suť, naplánuje její těžbu. Když má agent u sebe nějaký nástroj, nejprve nástroj použije v místě cílové akce, čímž dokončí plán, nebo nástroj vrátí do základny a plán zruší. Rozhoduje se na základě délky cesty, kterou by musel vykonat od jeho aktuální pozice k místu se sutí přes základnu nebo přes cílové místo původního plánu. Když nezná cestu k cílovému místu, vždy vrací nástroj do

základny. Potom následuje těžba. Jestliže žádný nástroj u sebe nemá, na políčko se sutí se přesune okamžitě. Vytěží jeden kus suti a odveze ho uložit do základny. Těžbu suti opakuje do jejího vyčerpání, potom se teprve pustí do nového plánu.

Oprava

Vykonání plánu na opravu své věže má přednost před útokem na cizí věž. Agent se rozhodne opravit svoji věž jen v případě, že existuje nějaká jeho poškozená věž a v základně je alespoň jedna cihla nebo suť. Věž považuje za poškozenou a vhodnou k opravě, jestliže hodnota života věže klesne pod určitou hranici. Tato hranice je rovna hodnotě života věže, na kterou bylo pouze dvakrát zaútočeno. Je tedy možné, aby věž byla ještě minimálně dvakrát opravena.

Nejprve se agent přesune na základnu, kde si vezme cihlu. V případě, že na základně žádná cihla není, jednu ze suti vytvoří. S cihlou potom jde k nejbližší poškozené věži, kterou opraví. Jestliže se v průběhu vykonávání plán stane nesplnitelný (někdo věž zničí nebo opraví na maximum života), musí plán změnit. Nenese-li ještě cihlu, plán ruší a začíná nový. Když už cihlu má, pokusí se najít nějakou svou věž, která potřebuje opravit, a případně plán podle toho změní. Neexistuje-li žádná poškozená věž, vrátí cihlu do základny a začne nový plán.

Útok

Nemá-li agent nic důležitějšího na práci, zvolí si plán, ve kterém zaútočí na cizí věž. Začíná tím, že jde na základnu, kde se vybaví zbraní. Jestliže na základně žádná zbraň není, musí nejprve nějakou vyrobit. Když má zbraň, vybírá věž k útoku. Na rozdíl od svých věží zná cestu ke všem cizím věžím, u kterých zná jejich polohu, protože je musel už dříve objevit. K útoku si vybírá věž, k níž to má nejblíže. Nezná-li polohu žádné cizí věže, pokusí se nějakou najít a na tu potom zaútočit. V případě, že se cíl plánu v průběhu jeho vykonávání stane nedosažitelný (někdo věž zničí), postupuje obdobně jako při opravování, ale s jediným rozdílem. Nezná-li žádnou další věž, zbraň nevrací, ale hledá další neobjevenou soupeřovu věž.

Manévr

V případě, že agent přijde do okolí soupeřovi budovy, na které má provést akci, a tato budova je soupeřovým týmem zablokována, musí provést manévr, aby ji soupeř přestal hlídat. V tomto režimu si nemůže zavolat žádnou podporu. Proto opustí okolí budovy a zůstane vyčkávat hned za jeho okrajem. Počká několik kol (dobu, kterou by agent stojící na budově potřeboval k opuštění jejího okolí) nebo zkusí najít budovu stejného typu ve vzdálenosti, která je srovnatelná s dobou čekání na odblokování věže. Pokud se rozhodne počkat, přesune se po uplynutí doby jen o jedno políčko do okolí budovy a zjistí její stav. V případě, že je budova stále zablokována, zdvojnásobí přijatelnou vzdálenost pro výběr jiné budovy a pokusí se najít náhradní cíl. Jinak opakuje podruhé čekací cyklus. Jestliže je budova i po druhém čekání stále obsazena, vybírá jinou nejbližší budovu, kde lze provést stejnou akci. Pokud taková budova není, vrací nástroj do základny, ruší plán a začíná nový.

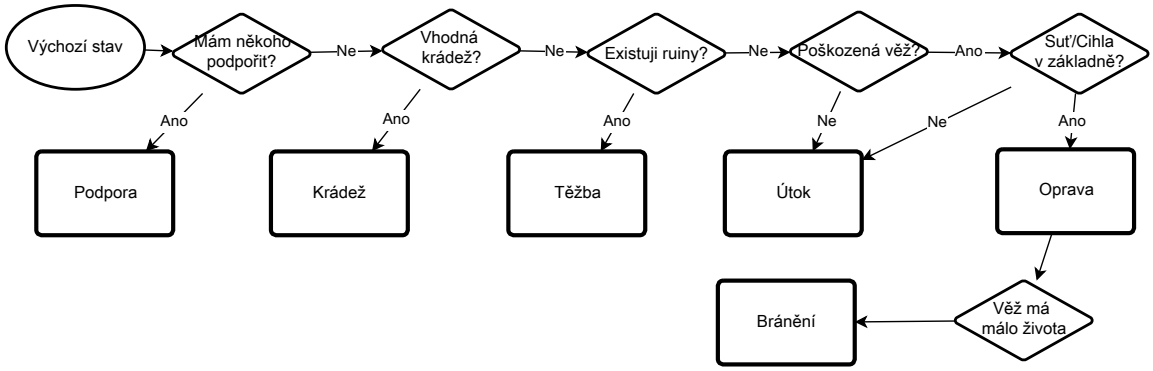
Krádež

Tento režim inteligence nikdy nebude zkoušet krádež z cizí základny, protože z hlediska blokování budov musí být v okolí základny o dva útočné agenty víc než obranných. To znamená, že je ke krádeži potřeba minimálně dvou agentů. V tomto režimu agenti nekomunikují a nemohou se tedy domluvit na společné krádeži. Proto by byla jen náhoda, kdyby se v okolí sešlo více agentů současně a počkalo dostatečnou dobu na provedení krádeže, na což nelze spoléhat.

6.3 UI2 – mírná spolupráce

Tento režim inteligence vychází z režimu předchozího. Jsou zachovány některé základní rozhodovací prvky, ale je složitější, protože agenti v týmu komunikují a částečně spolupracují. Neplánují dopředu spolupráci s ostatními agenty, pouze žádají o jejich podporu, kterou aktuálně potřebují.

V tomto režimu, stejně jako v tom předešlém, můžeme činnost agenta rozdělit do několika dlouhodobějších plánu. Je jich ale o tři více, tedy šest. Kromě těžby suti, opravy vlastní věže a útoku na cizí věž si může agent navíc naplánovat okrást nepřátelskou základnu, bránit vlastní věž nebo podpořit jiného agenta svého týmu. Převzaté plány mají stejné jméno a cíl, ke kterému vedou, ale některé části jsou prováděny odlišně s využitím vzájemné spolupráce mezi agenty.



Obrázek 6.2: Základní princip výběru dlouhodobého plánu agenta UI2.

Komunikace

Komunikují spolu pouze agenti ze stejného týmu. Komunikace mezi agenty je dvojího typu: žádost o podporu nebo předání informací o sobě a prostředí. Zatímco informace předává všem agentům svého týmu vždy, když zjistí něco nového, o podporu agent žádá jen určité agenty, a to jen když ji potřebuje.

O prostředí si agenti předávají pouze statické informace — rozložení terénu a umístění budov. Tyto informace si posílají ihned po tom, co objeví nové políčko. Také si navzájem zasílají informace o svém aktuálním stavu: polohu, držený nástroj, právě prováděný plán a případnou polohu místa, které mají podpořit. O každé jejich změně se okamžitě informují. Každý agent tedy zná polohu a plán všech ostatních v jeho týmu.

Podpora

Novým principem v tomto režimu inteligence je žádost o podporu. O podporu žádá agent, jenž má provést nějakou akci na zablokované budově. Cílem je přivolat do okolí této budovy své agenty, aby tak získali převahu a umožnili provedení akce. Agent, který přijde do okolí zablokované budovy, na níž má provést akci, se nejprve chová stejně jako v předchozím režimu. Pokusí se v blízkosti najít náhradní cíl. Jestliže ho nenajde, vyčkává ze okrajem okolí budovy několik kol, aby měli cizí agenti dostatek času okolí opustit. Po čekání se vrátí do okolí. Není-li budova dále zablokována, provede akci a končí plán.

V případě, že je budova stále zablokována, stává se z agenta iniciátor. Iniciátor je agent, jenž řídí podporu. Jedna budova má v jeden čas maximálně jednoho iniciátora. Iniciátor si pamatuje všechny agenty, kteří ho mají podpořit, a po dokončení plánu ruší veškeré své požadavky na podporu. Nikdy podporu nezruší, dokud není plán splněn nebo není rozhodnuto o jeho neúspěchu.

Iniciátor žádá o podporu své kolegy postupně. Nejdříve počká, až přijde do okolí budovy jeden podporující agent, a až poté, v případě potřeby, volá dalšího. Po zavolání podpory se iniciátor přesune na budovu, kde čeká na příchod svých agentů do jejího okolí. Dále volá potřebnou podporu a vyčkává do okamžiku, než jeho tým získá na budově převahu, čímž mu odblokuje budovu a umožní akci. Ihned akci provede, čímž dokončí plán, a následně odvolá podporu. Když při jakékoli žádosti o podporu nemá iniciátor k dispozici volného agenta, veškerou podporu ruší a odebírá se mu funkce iniciátora. Dále hledá nejbližší vhodnou budovu jako náhradní cíl plánu. Neexistuje-li taková budova, vrátí nastroj do základny a začne jiný plán.

Vykonávání podpory se nesmí přerušit nebo pozdržet jiným plánem, dokonce ani jinou podporou, protože na ni minimálně jeden agent čeká. O podporu lze tedy zažádat jen agenty, kteří právě nemají žádnou jinou naplánovanou a nejsou iniciátory. Při výběru vhodného agenta k podpoře se volní (nejsou podpora) agenti pomyslně rozdělí do následujících skupin:

- agenti blízko budovy,
- agenti s plánem, který má stejné místo cíle, jako je budova,
- agenti bez dlouhodobějšího plánu,
- agenti se stejným typem plánu jako iniciátor, který ale mají vykonat na jiném místě,
- ostatní volní agenti.

Jako podpora se vybere agent z první neprázdné skupiny, dle výše uvedeného pořadí. Ve skupině se zvolí agent, který je nejblíže zablokované budově.

Podpora probíhá tak, že agent dostane souřadnice budovy, v níž potřebuje pomoc jiný agent. Okamžitě pozastaví vykonávání aktuálního plánu a po nejkratší cestě se přesune do jejího okolí. Zůstane stát na okraji okolí budovy, kde čeká dokud mu iniciátor podporu nezruší. Potom se agent vrací ke svému původnímu plánu.

Agent, který v rámci svého vlastního plánu plní také funkci podpory, nečeká na okraji okolí. Nepokouší se ani o čekání mimo okolí, jak to dělají agenti, než se z nich stanou iniciátoři. Okamžitě se přesune na budovu, kde čeká na její odblokování, aby mohl ihned provést akci.

Oprava, útok a těžba

Plány na těžbu suti, opravu a útok věže jsou z částečně stejné jako u předchozího režimu, stejně jako jejich výběr. Agent se nejdříve zbaví nevhodného nástroje nebo získá ze své základny nástroj požadovaný. Potom vybere nejbližší vhodnou věž a jde k ní. Když není věž zablokovaná, akci provede. V tomto případě je chování agenta stejné jako v předešlém režimu. Odlišné začne být v případě, že je cizí věž zablokovaná. Potom se agent stává iniciátorem a volá podporu, jak je popsáno výše.

Krádež

Na rozdíl od předchozího se v tomto režimu mohou agenti domluvit a uskutečnit krádež. Jedná se o jediný plán, v němž se využívá hromadné podpory. Od klasické podpory se liší tím, že je možné zavolat k podpoře jedné budovy více agentů současně, a jejich výběrem. Zde si může iniciátor dovolit zavolat k podpoře více agentů a tím zdržet jejich vlastní plán, protože vybírá pouze agenty, kteří momentálně neplní funkci podpory a jsou od soupeřovi základny vzdáleni maximálně trojnásobek jejich dohledu.

Krádež v soupeřově základně agent naplánuje, když prochází jejím okolím, a může tak zjistit její aktuální stav. Původní plán zruší a nahradí krádeží, jenom když nenese žádný nástroj, je v základně cihla nebo suť a je v blízkosti k dispozici dostatek jiných agentů, kteří pomohou s odblokováním základny.

Podle množství soupeřových agentů v okolí základny iniciátor spočítá, kolik potřebuje minimálně svých agentů, aby budova nebyla zablokovaná. Požádá tento minimální počet kolegů o hromadnou podporu a vyčkává na soupeřově základně. Po odblokování základny okamžitě ukradne cihlu nebo suť. Je zbytečné a nevýhodné krást zbraň, protože si ji může jednoduše sám vyrobit. Po krádeži se agent pokusí vykonat další plán podle nástroje, který zrovna vzal, jinak jej odnese do své základny.

Jestliže přijdou do okolí základny všichni agenti, které k podpoře povolal a základna stále není odblokovaná, zavolá podle potřeby další hromadnou podporu. Nemá-li iniciátor dostatek agentů potřebných k odblokování, které by požádal o hromadnou podporu, plán ruší stejně jako veškerou podporu. Stejně tak se plán stane nesplnitelný, když na soupeřově základně dojdou cihly a suť.

Bránění

Plán bránění své věže spočívá v tom, že agent zůstane v jejím okolí, dokud ji jiný agent neopraví, i když nejsou v jejím okolí žádní cizí agenti. Tento plán agent spouští, jen když právě opravil věž a životy věže nepřesáhly určitou hranici (stačí dva útoky do zničení věže). Aby agent věž nebránil zbytečně, musí být zajištěno, že jí přijde jeho kolega opravit. Buď už má nějaký agent v plánu věž opravit, nebo je v základně suť nebo cihla, kterou si pravděpodobně další volný agent vezme a půjde věž opravit. Jestliže věž nemůže nikdo opravit, věž nebrání.

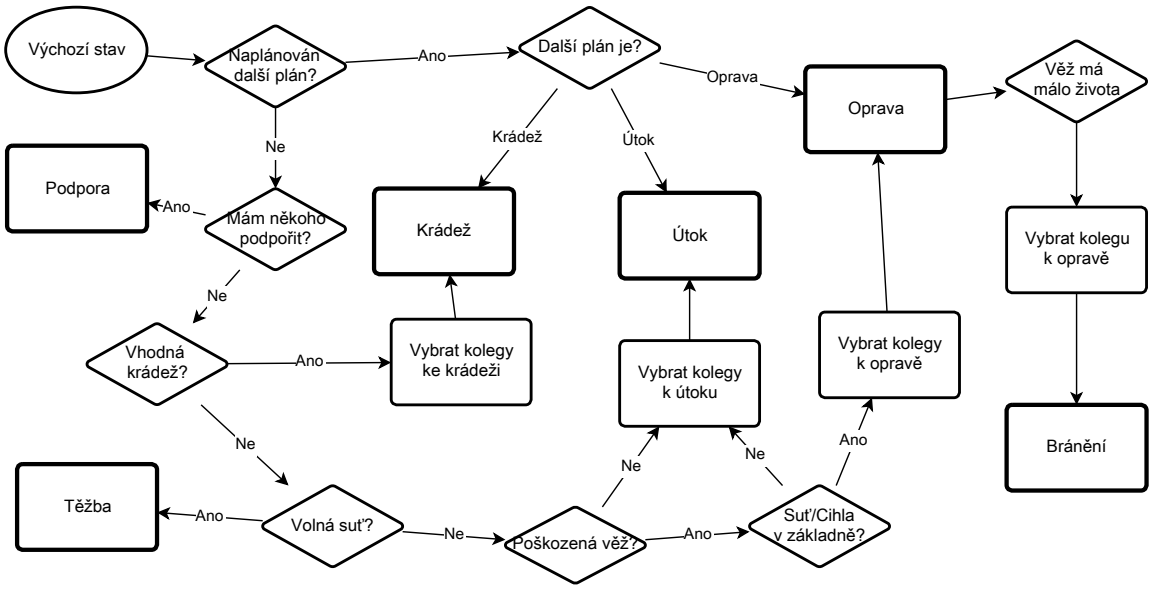
Bránící agent čeká na okraji okolí věže, na políčku nejblíže své základny, kam se potom vydá. Vyčkává, dokud se někdo nevydá věž opravit. Agenta, jenž má v plánu věž opravit, požádá bránící agent o podporu, čímž mu nastaví příznak podpory, aby jej nikdo nemohl přerušit. Potom čeká na příchod opraváře a mezitím nastavuje příznak podpory dalším agentům, kteří mají v plánu také věž opravit. Dělá to, protože věž nutně potřebuje opravit a ostatní mohou mít výhodnější polohu než původní opravář. Jakmile přijde opravář do okolí věže, bránící agent končí obranu a opouští okolí věže, aniž by počkal na provedení

opravy. Jestliže je budova zablokovaná a je třeba přítomnost bránícího agenta v jejím okolí, opravující agent ho volá jako normální podporu. Pokud soupeř i přes agentovo bránění věž zničí, plán pro něj automaticky končí, stejně jako pro agenty, kteří měli věž opravit.

6.4 UI3 – komplexní spolupráce

Tato úroveň inteligence je nejpropracovanější a je v ní kladen největší důraz na vzájemnou spolupráci mezi agenty. Vychází z předchozího režimu, ale na rozdíl od něj agenti dopředu plánují vzájemnou spolupráci a skupinové provádění některých plánů. Agenti jednoho týmu si také navíc mezi sebou předávají veškeré informace.

Tento režim má stejné dlouhodobé plány jako režim mírné spolupráce. Tyto plány by se dali rozdělit do dvou skupin podle způsobu jejich plnění. První skupinou jsou plány týmové, kam patří útok na cizí věž, oprava vlastní věže a krádež v nepřátelské základně. Tyto plány jsou současně plánovány dopředu pro více agentů, kteří se jich mají společně zúčasnit a tím je zefektivnit. Ve druhé skupině jsou plány individuální. Je to těžba suti, obrana budovy a podpora ostatních agentů. Nejsou dopředu plánovány ani společně vykonávány. Může sice probíhat stejný individuální plán pro více agentů současně, ale navzájem se přímo neovlivňují.



Obrázek 6.3: Základní princip výběru dlouhodobého plánu agenta UI3.

Komunikace

Komunikace v tomto režimu je rozsáhlejší než v tom předešlém, ale funguje na stejném principu. Kromě žádostí o podporu a předávaní informací si agenti navíc navzájem plánují svojí další činnost.

Agenti si předávají veškeré informace, které o prostředí mají. Tedy kromě statických také dynamické, což jsou zdraví budov, počet nástrojů a stav blokovaní budov. Kdykoliv agent zjistí nové aktuální informace nebo potvrdí trvání dynamických, okamžitě je pošle všem agentům v týmu. Agenti, kteří tyto informace obdrží, si je uloží nebo podle nich aktu-

alizují své dosavadní představy. U každého údaje zaznamenají také čas, kdy byli z prostředí zjištěny, z čehož lze později odvodit jejich aktuálnost.

Stejně jako v předchozím režimu si agenti předávají informace o svém aktuálním stavu, které po každé změně všem rozesílají. Oproti předchozímu režimu přibyly údaje potřebné pro týmové plánování. Navíc se tedy agenti informují o všech svých aktuálních či budoucích plánech, včetně předpokládaného místa a času jejich ukončení.

Týmové plánování

Hlavní výhodou tohoto režimu by mělo být vykonání týmových plánů, což je krádež, útok a oprava věže. Týmový plán je takový, který může plnit více agentů současně a tím zefektivnit celkovou činnost týmu. Při plnění tohoto plánu postupují agenti ve skupině, což jim dává lokální výhodu při blokování budov. Tím se snižuje pravděpodobnost potřeby podpory, která vede ke zdržení podporujícího agenta od svého plánu.

Týmové plány jsou pro celou skupinu vytvořeny jedním agentem — vůdcem. Vůdcem se stane agent, který nemá žádný další plán a rozhodne se pro jeden z týmových plánů. Pro týmový plán vůdce vždy vybírá ke spolupráci jen agenty, u nichž předpokládá, že se jim podaří splnit tento plán v rozmezí deseti kroků od splnění jím samotným. Tito agenti také nesmí zrovna plnit funkci podpory, protože to zdržuje jejich vlastní plán a jeho předpokládané ukončení se může značně lišit od skutečnosti. Rozhodnutí o vhodných spolupracovnících provádí vůdce na základě předpokládaných údajů o čase a místě ukončení poslední naplánované činnosti a délky cesty, jakou bude muset urazit, případně bere v úvahu také potřebnou cestu přes základnu a manipulaci s nástrojem. Z čehož plyne, že v ideálním případě, kdy nedojde ke zdržení žádného agenta, bude při provedení konečné akce celá jeho skupina nedaleko okolí budovy.

Po tom, co vůdce naplánuje další činnost celé vybrané skupině a sobě, se dál o průběh a vykonávání plánu ostatními agenty nestará. Další jejich činnost nemusí nijak řídit, protože všichni plán znají a vědí jak reagovat na jeho neúspěch. Vůdce totiž nemůže hromadně celé skupině změnit cíl plánu, protože není zaručeno, že agenti ve skupině budou náhradnímu cíli vyhovovat. Je tedy lepší vytvořit skupinu novou. V případě, že plán nelze splnit, vědí o tom všichni agenti ze skupiny, protože mají stejné informace o prostředí. Na což reaguje každý agent individuálně, podle vlastní situace. Buď plán zruší, nebo najde jeho alternativu na jiném místě, čímž se z něj stane vůdce jiné skupiny, kterou nově vytvoří.

Oprava

Týmový plán na opravu vlastní věže si vůdce vybere, jen když existuje poškozená věž, jíž nemají jeho kolegové v plánu úplně opravit. Přednost před tímto plánem má pouze těžba suti a krádež. Vůdce si zjistí, kolik má dostupných agentů k vykonání tohoto týmového plánu. Potom spočítá, kolik je k dispozici cihel a kolik jich může ze suti vyrobit. Od tohoto množství odečte počet nevybavených agentů, kteří mají v plánu se cihlou vybavit, čímž získá počet volných cihel, jež může využít. Zná tedy maximální počet dostupných agentů, které je schopen vybavit. Najde si nejbližší poškozenou věž, kde by mohl využít maximální množství dostupných agentů, přičemž vychází z aktuální hodnoty života věže a počtu jiných agentů, kteří už mají v plánu tuto věž opravit. Na základě těchto dedukcí a výpočtů vůdce zaúkoluje maximální počet vhodných agentů, jež využije k opravě. Dále plní agenti plán, stejně jako v předchozím režimu.

Útok

Plán útoku na cizí věž je založen na stejném principu, jako ten na opravu, jen s pár rozdíly. Vůdce je schopen vždy vybavit zbraněmi všechny dostupné agenty. Snaží se vybrat pro hromadný útok nejvhodnější věž tak, aby po útoku všech členů skupiny byla věž pokud možno zničena. Stejně jako u opravy vůdce nikdy neúkoluje více agentů, než kolik pravděpodobně využije ke zničení věže. Vychází z poslední zjištěné hodnoty života věže a počtu jiných agentů, kteří mají v plánu na tuto věž zaútočit. Při plnění tohoto plánu agenti postupují stejně jako v předchozím režimu.

Krádež

Na rozdíl od předchozího režimu, kdy se na pomoc s krádeží volá hromadná podpora, se v tomto režimu krádež vykonává jako týmový plán, jenž je výhodnější. Pro plán krádeže z cizí základny se vůdce rozhodne, pokud má informace o této základně, které nejsou staré více než deset kol a má k uskutečnění týmovému plánu dostatek volných agentů, jež odblokují základnu. Aby se krádež vyplatila, musí být pro každého agenta, jenž se jí účastní, v soupeřově základně suť nebo cihla ke krádeži. Jsou-li tyto podmínky splněny, vůdce naplánuje, co největšímu počtu vhodných agentů krádež. Dále už agenti pokračují v plánu stejně jako v předchozím režimu.

Bránění

V tomto režimu inteligence je plán bránění věže zvláštním typem plánu. Nejedná se o čistě individuální plán ani o plán týmový ale o něco mezi nimi. Tento plán je stejný jako v předchozím režimu, avšak s jediným podstatným rozdílem. V případě, že nikdo nemá v plánu věž opravit, nemusí bránící agent doufat, že se k tomu někdo časem rozhodne, ale přímo naplánuje tuto opravu jinému agentovi. Vybírá volného agenta, u kterého předpokládá, že opraví věž nejrychleji.

Těžba

Tento plán je také stejný jako v předchozím režimu, ale než se agent pro něj rozhodne, bere v úvahu činnost svých kolegů. Tento plán si zvolí, pouze pokud je mu k dispozici volná suť. Volná suť představuje kus suti, jehož těžbu nemá žádný kolega v plánu, neboli suť, která zbude po vykonaní aktuálních těžebních plánů všech kolegů. Volná suť nezaručuje úspěch těžby, protože ji může mezitím vytěžit soupeř, ale její nedostatek dokazuje jistý neúspěch těžby, neboť na agenta žádná suť určitě nezbude. Nikdy si tedy těžbu na jednom místě nenaplánuje více agentů jednoho týmu, než je dostupné množství suti.

Podpora

Princip a funkce podpory je stejná jako v předchozím režimu. Liší se jen rozdílným přístupem k výběru agenta, z něhož se má stát podpora. Pomyslné rozdělení do skupin je zachováno stejně jako skupiny a jejich pořadí. Odlišný je výběr vhodného agenta ze skupiny. V předchozím režimu se ze skupiny vybírá agent, jenž je nejblíže, aby jeho zdržení a čekání na něj bylo co nejkratší. V tomto režimu se musí u agenta, který vykonává týmový plán, navíc zohlednit jeho zdržení, jenž může způsobit zdržení ostatních agentů se stejným plánem, kteří by na něj museli čekat nebo volat podporu. Proto se u těchto agentů při výběru nejvhodnější podpory, neuvažuje vzdálenost od místa podpory, ale předpokládané zdržení vykonaní jeho týmového plánu.

Kapitola 7

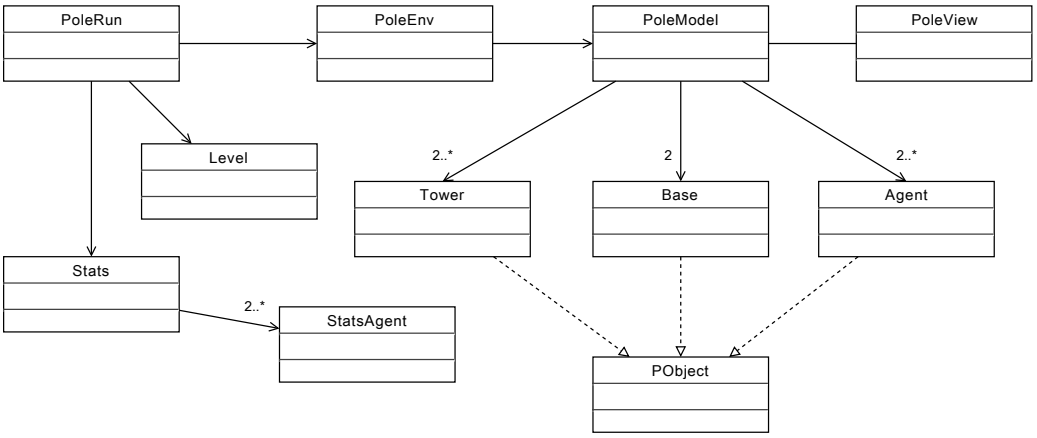
Implementace

V této kapitole je popsána implementace výše navrženého multiagentního systému. Tento systém se skládá ze tří odlišných modulů, které spolupracují. První modul řídí celou simulaci. Jeho jádrem je kompletní model prostředí, se kterým manipuluje. Dále se stará o vizualizaci a sběr statistik. Ve druhém modulu jsou implementovány všechny režimy inteligence, tedy chování agentů a jejich rozhodovací proces. Poslední modul je nejmenší a obsahuje pouze několik speciálních funkcí, které agenti při své činnosti využívají. Každému modulu je níže věnována samostatná podkapitola.

Tento systém je realizován ve frameworku Jason, viz kapitola 3.3. Jason spravuje běh celého sytému a obstarává spolupráci jednotlivých modulů. Více informací, včetně samotného frameworku a jeho zdrojových kódů, je dostupných na webu [9].

7.1 Model prostředí

Tento modul je implementován v jazyce Java a vychází z návrhu prostředí a úlohy v kapitole 5. Je složen z několika tříd, jež jsou schématicky znázorněny na obrázku 7.1, včetně jejich vzájemných vztahů. Název stěžejních tříd obsahuje slovo *Pole*, které jsem původně zvolil jako provizorní jméno aplikace, jež jí už ale zůstalo. Dále bude popsána funkce těchto důležitých tříd, ostatní pomocné třídy jsou rozebrány v příloze D.1.



Obrázek 7.1: Objektový diagram prvního modulu — model prostředí.

PoleRun

Tato třída obaluje a inicializuje celou simulaci. Obsahuje všechny důležité konstanty, které se v tomto modulu používají, a také obstarává výpis všech statistik.

Výsledná aplikace se spouští přes tuto třídu, jež zpracuje zadané parametry (viz příloha B). Na základě těchto parametrů načte mapu a vytvoří inicializační soubor simulace ve formátu *mas2j*, který Jason používá. Tento soubor určuje režim inteligence a počet agentů v týmech, také udává třídu, která reprezentuje prostředí simulace, což je `PoleEnv`. Samotnou simulaci spustí, když název tohoto souboru předá jako argument funkci `main` ze třídy `RunCentralisedMAS`, jež je součástí Jasonu. Tato třída dle zadaného souboru spustí a spravuje běh multiagentního systému, v němž jsou činnosti všech agentů řízeny z jednoho místa.

PoleEnv

Instance této třídy reprezentuje prostředí a řídí běh simulace. Aby mohl Jason spravovat běh systému a spolupráci jeho částí, musí tato třída vycházet z připravené třídy `Enviroment`. Moje třída je odvozená od jejího potomka a to ze třídy `TimeSteppedEnvironment`. Tato další připravená třída navíc synchronizuje akce všech agentů, rozdělí tedy simulaci na kola, ve kterém může agent provést pouze jednu akci. Dělá to tím způsobem, že se těsně před tím, než agent provede akci, zastaví jeho činnost. Čeká, než se do stejné situace dostanou všichni agenti a až potom je všechny nechá akci vykonat. Jakmile všichni agenti akci dokončí, obnoví jejich činnost.

Jason automaticky vytváří instanci této třídy na základně inicializačního souboru a řídí podle ní běh simulátoru. Ale v první řadě se v rámci její inicializace vytvoří a inicializuje model prostředí a případně i jeho grafická reprezentaci.

V systému plní objekt této třídy také roli prostředníka mezi agenty a modelem prostředí. Pomocí funkce `executeAction` deleguje akce agenta na příslušné metody modelu prostředí, které na něm vykonají požadovanou změnu. Mimo to také informuje agenty pomocí připravených funkcí o aktuálním stavu prostředí. V každém kole všem agentům předá vybrané informace — vjemy, což vždy provede po vykonání akcí všech agentů, ale ještě před obnovením jejich činnosti.

Jako jediný má tento objekt přehled o stavu simulace a počtu proběhnutých kol. Proto po každém kole před předáním vjemů volá metodu modelu, která jej aktualizuje. Také po skončení simulace doplní statistiku a dá pokyn k jejímu vyhodnocení a vypsání.

PoleModel

Model prostředí a jeho aktuální stav je po celou dobu simulace reprezentován jedním objektem této třídy. Stav prostředí je uvnitř tohoto objektu uložen pomocí několika polí a seznamů.

Model obsahuje tři dvourozměrná pole typu `boolean` o velikosti mapy, kde každá položka udává stav jednoho políčka. Nejdůležitější je pole s rozvržením terénu, ve kterém jsou označena políčka, na nichž je zeď. Další dvě pole jsou jen doplňková a slouží ke zvýšení efektivity. První z nich označuje místa na mapě, kde stojí nějaká budova, což zrychluje jejich vyhledávání. V druhém poli, které se na rozdíl od předchozích dynamicky mění, se označují políčka, na nichž došlo v tomto kole ke změně obvykle způsobené přesunem agenta. To odlehčuje vykreslování, protože se nemusí překreslovat celá mapa, ale jen změněná políčka.

Pro reprezentaci konkrétních budov a agentů včetně jejích kompletních stavů využívá model dalších objektů. Tyto objekty jsou, kvůli usnadnění a urychlení manipulace, sdružovány do polí a seznamů podle typu a týmu. Jedná se o objekty pomocných tříd, které jsou samostatně popsány.

Model dále obsahuje spoustu nejrůznějších metod, pomocí kterých může zjišťovat a měnit svůj stav. Tyto metody lze podle funkce rozdělit do čtyřech větších kategorií. V první jsou metody zabezpečující přidání nových prvků do modelu, které se používají zejména při jeho tvorbě. Další skupinu tvoří metody pro vyhledání objektů, jež vrací seznam budov nebo agentů zvoleného týmu, které jsou v zadané oblasti. Ve třetí kategorii jsou informační metody pro usnadnění práce s mapou. Poslední skupina obsahuje metody, které odpovídají akcím agentů. Agenti pomocí nich mění model prostředí a snaží těmito změnami dosáhnout svého cíle. V těchto metodách se ještě před samotným provedením změny ověřuje na základě aktuálního stavu prostředí, jestli jsou splněny všechny podmínky k jejich vykonání, jinak se požadovaná akce neprovede.

Na začátku simulace se model prostředí vytvoří na základě objektu třídy `Level`, jenž obsahuje informace o počátečním rozložení všech prvků na mapě.

Výše zmíněná metoda na aktualizaci modelu, kterou po každém kole volá `PoleEnv`, slouží k přepočítaní stavu blokování budov a k případnému překreslení vizualizace tohoto modelu. Počet agentů v okolí budovy se mohl od minulého kola změnit a neexistuje žádný mechanismus, jenž by ji o tom informoval. Proto musí všechny budovy svůj stav po každém kole přepočítat na základě aktuálních pozic agentů.

PoleView

Instance této třídy slouží k vizualizaci aktuálního stavu modelu prostředí. Tato třída je odvozená z třídy `JFrame` a k vykreslování používá objekt třídy `Canvas`.

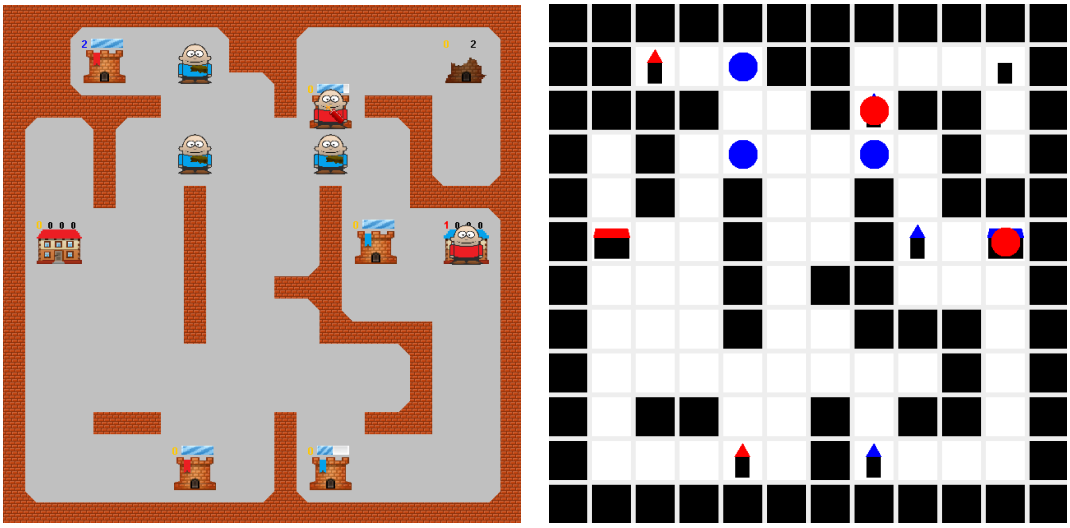
Zobrazení této grafické reprezentace modelu je volitelné a výsledek simulace nijak neovlivňuje. Prodlouží jen celkovou dobu jejího běhu o čas potřebný k překreslení. Jak už bylo sděleno výše, v každém kole po proběhnutí akcí všech agentů se překreslí pouze políčka, která byla změněna, a všechny budovy, u nichž se stav mění skoro po každém kole.

Tato třída obsahuje pro každý prvek prostředí funkci k jeho samostatnému vykreslení, jež se přizpůsobí jeho stavu. O políčku, jenž chce překreslit, si pomocí metod modelu nejprve zjistí, jaké prvky obsahuje. Potom podle nich volá vhodné metody. Nejprve vykreslí terén, potom budovu a až nakonec agenty, kteří jsou na zvoleném políčku.

Třída obsahuje dvě různé podoby vizualizace. Jednu propracovanější s více detaily, v níž se zobrazují nástroje, které agent právě nese, počty nástrojů uložené v základně, stav života věže nebo počet suti. U každé budovy se také zobrazuje, který tým má v jejím okolí převahu a jak velkou. Zobrazení detailů u objektů je dosaženo použitím jejich bitmapového obrázku, jehož velikost se přizpůsobí velikosti políčka. Obrázky vycházejí z volně použitelných ikon[1], jež jsem pro svou potřebu upravil. Zeď je vytvořená pomocí, textury, která vyplní její tvar. Tvar zdi se počítá na základě rozložení zdi v sousedních políčkách tak, aby plynule navazovala při jakémkoli rozložení terénu. Tato podoba je tedy vhodná pro sledování průběhu simulace, kdy zobrazení většího množství údajů o prostředí pomáhá k pochopení činnosti agentů.

Druhá grafická podoba je naopak hodně strohá a bez detailů. Jedná se pouze o schématické zobrazení rozvržení terénu, pozic agentů a budov. Nezobrazuje žádné další údaje,

[1] http://www.iconarchive.com



(a) Detailní (b) Schématické

Obrázek 7.2: Srovnání různých zobrazení stejného stavu prostředí.

a proto je vhodná pro zobrazení výchozího rozložení prvků v prostředí. Srovnání těchto dvou grafických vzhledů je na obrázku 7.2, kde oba zachycují stejný stav prostředí.

7.2 Inteligence

V tomto modulu je implementováno chování agentů, které je navrženo v kapitole 6. Tento rozhodovací proces agentů je vytvořen přímo v Jasonu, tedy přesněji v jeho rozšířené verzi jazyka AgentSpeak(L). Jason tento kód interpretuje, čímž vykonává činnost agenta a deleguje jeho akce na model prostředí.

Každý režim inteligence má svůj soubor (`aui1.asl, aui2.asl, aui3.asl`), jenž obsahuje jádro jeho rozhodovacího procesu. Kromě toho využívají všechny režimy další čtyři soubory (`rules.asl, move.asl, move.asl, stats.asl`), do kterých jsou podle souvislostí rozděleny mechanismy a činnosti, které jsou pro režimy společné. Stručná charakteristika těchto souborů, včetně jejich podstatných prvků se nachází v příloze D.2.

V této kapitole bude nejprve popsán základní přístup k řízení činnosti agentů a předávání informací. Dále budou uvedeny a zdůvodněny odchylky implementace oproti návrhu a několik přidaných principů.

Řízení činnosti agentů

V multiagentním systému, který je realizován v Jasonu, se při řízení činnosti agentů mezi nimi náhodně přepíná. Agenti si tak navzájem pozastavují vykonávání své činnosti. Tento princip je pro mě nevhodný, protože potřebuji změřit, jakou dobu trvá jednotlivému agentovi kompletní výpočet následující akce. Proto jsem se rozhodl vynutit u agentů pevné pořadí, ve kterém po jednom postupně vykonají celý rozhodovací proces. To znamená, že s výpočtem další akce agent začne, až jej dokončí jeho předchůdce. Toto umožní celkem přesně změřit čas mezi začátkem a koncem agentova výpočtu. Mohu zanedbat velmi krátkou dobu, kterou bude trvat přepnutí na jiné agenty a okamžitý návrat zpět, protože jsou všichni ostatní agenti pozastaveni a nemůžou nic vykonat.

V reakci na toto vynucené pořadí je třeba vyřešit problém, kdy agent má nějaké podstatné informace, které by výrazně ovlivnily rozhodnutí jeho kolegy, jenž se rozhoduje ještě před tím, než je agent vůbec bude moct zaslat. Kolega je ale vezme v úvahu až v dalším kole, kdy už můžou být zcela chybné. To by mohlo vést k těžko odhalitelným informačním konfliktům mezi agenty a narušit tak činnost celého týmu. Proto jsem se rozhodl předat všechny nové informace na začátku kola ještě před zahájením výpočtu prvního agenta. Aby se toto zajistilo, agenti se informují postupně v pevném pořadí, stejně jako při výpočtu další akce. Výhodou tohoto přístupu je také celkové snížení množství zaslaných zpráv, protože se informace neposílají po jedné, ale hromadně v jedné zprávě. V režimu bez spolupráce se žádné informace nezasílají, ale agenti musí tuto, pro ně prázdnou, fázi vykonat, aby se zachovalo jejich pořadí. V ostatních režimech se měří doba, kterou jednotlivým agentům zabere vlastní informační fáze.

Každé kolo se tedy skládá ze dvou částí, a to z fáze informační a fáze výpočetní neboli rozhodovací. Fáze rozhodovací začne až po tom, co poslední agent ukončí fázi informační, v době kdy mají všichni agenti v týmu shodné aktuální informace.

Postupná činnost agentů je řízena předáváním pomyslné štafety, kterou si agenti předávají pomocí zaslání zprávy. Po vykonání jedné fáze předá agent štafetu následníkovy a sám čeká na štafetu ve druhé fázi. Agent čeká na štafetu pomocí funkce `.wait`, kterou volá hned po vykonaní akce, před tím než začne s jakoukoli činností. Tato funkce zastaví jeho činnost a znovu jí spustí, až obdrží zprávu se štafetou.

Tyto zprávy, jimiž se řídí předávání štafety, se nezahrnují do statistiky, protože jsou pouze pomocné a jejich počet je vždy stejný. Celková doba simulace je díky této postupné dvoufázové činnosti značně delší, což ale není moc podstatné, protože se měří pouze čistý čas, jenž agent stráví aktivní činností v obou fázích. Přičemž tento čas bude obdobný jako součet časů, které by při přerušovaném běhu strávil tento agent nad stejnou činností. Tyto dílčí časy by šlo zjistit jen velmi těžko a stejně nepřesně, protože přepínání mezi agenty řídí Jason, jenž tyto časy neměří.

Rozdíly oproti návrhu

Většina implementace chování agenta dodržuje základní rysy a principy návrhu, ale v průběhu implementace jsem se místy od návrhu odklonil, protože jsem přišel na výhodnější řešení.

V prvotním návrhu měli agenti volat podporu po skupinách, a to tak, aby velikost této skupiny pružně reagovala na aktuální stav zablokované budovy. Iniciátor měl tedy při každé změně počtu agentů v okolí budovy odvolávat část podpory nebo volat další. Vzhledem k blokování budov se jedná o velmi dynamické prostředí, které se mění prakticky s každým krokem, a proto jsem tento princip brzy zamítl.

Největším rozdílem oproti návrhu je chování agentů ze všech režimů, když je pro ně soupeřova budova zablokována a provádí manévr. Podle návrhu spočívá manévr v tom, že agent několik kol počká mimo okolí a pak se pokusí splnit plán znovu. Manévr je implementován odlišně. Agent nečeká, ale hned jakmile opustí okolí budovy, se do něj v dalším kole vrátí. Výhodou o proti návrhu je, že agent může rychleji reagovat na změny stavu budovy a nikdy zbytečně nečeká.

Agent v režimu bez spolupráce tento manévr opakuje a pokouší se svůj plán splnit. Jestliže je budova po vykonání pěti manévrů stále zablokována, vybere agent budovu jinou, kde by mohl vykonat stejný plán. Jinak plán končí neúspěchem.

Dostane-li se do stejné situace agent jiného režimu, chová se podobně. Také nečeká mimo budovu, ale okamžitě při prvním zjištění, že je budova zablokována, volá podporu.

Podle návrhu by se potom měl přesunout na budovu a počkat na podporu. Toto není zrovna nejlepší nápad, protože rozhodne-li se soupeř budovu hlídat, její okolí neopustí, pokud na ni bude cizí agent vyčkávat. Proto agent v době, kdy čeká na podporu, opakuje výše zmíněný nový manévr. Tedy opustí okolí budovy, hned se do něj vrátí a pokusí se uskutečnit plán znovu. V případě, že je budova stále zablokována, provádí tento manévr, dokud nepřijde jeho podpora. Dál pokračuje stejně jako v návrhu.

Navíc si agenti můžou vhodně upravit aktuální plán jako reakci na získání nové informace. Například když agent v průběhu vykonávání plánu objeví nové bližší místo, kde by mohl vykonat stejný plán, změní cílové místo plánu, a tak ho vykoná rychleji. Tento princip není využíván u týmových plánů, protože by se tím snížila efektivita společného plánu. Obrovskou výhodu přináší, když je požádán o podporu agent, jenž má stejný typ plánu jako iniciátor, ale akci má vykonat na jiném cílovém místě. Jakmile se dostatečně přiblíží, upraví svůj vlastní plán tak, aby provedl původní akci na místě, jenž měl jen podpořit. Tímto ušetří kroky, které by potřeboval k návratu k původnímu plánu, se zachováním jeho účinku.

Oproti návrhu je přidána další spousta drobných pravidel, díky kterým je v určitých situacích nalezeno výhodnější řešení, což vede k efektivnější činnosti agentů. Jsou to například tyto:

- Když se agent rozhoduje pro opravu věže, uvažuje, jestli blíž než věž, kterou má opravit, není velmi poškozená cizí věž, kterou by bylo výhodnější zničit.
- Jestli agent nemá nástroj a rozhoduje se mezi opravou a útokem, odloží své rozhodnutí až do příchodu na základnu, kam by stejně musel jít pro nástroj. Po příchodu se rozhodne podle aktuálních stavu prostředí.
- Agent, který nese nástroj a vybírá další plán, snaží ho zvolit tak, aby nástroj využil a nemusel jej vracet do základny.
- Zpracování veškerých zpráv provádějí agenti atomicky, aby zabránily jejich duplikaci.
- Při začátku nového plánu se automaticky odvolá veškerá podpora k plánu předešlému.
- V režimu komplexní spolupráce si agenti plánují maximálně dva plány dopředu, tedy aktuální a příští, protože je nepravděpodobné vhodně vybrat třetí plán, který se bude provádět za dlouhou doby, kdy bude prostředí ve zcela jiném stavu.

7.3 Speciální funkce

Tento modul obsahuje několik speciálních funkcí, které jsou taky označovány jako vnitřní akce agenta. Jason umožňuje vytvoření těchto vlastních funkcí, jež mohou agenti při své činnosti použít. Tyto funkce jsou implementovány v Javě. Obvykle se jedná o matematické funkce nebo funkce pro složitější algoritmické operace, které nejsou vhodné pro AgentSpeak(L).

Tyto funkce se implementují jako metoda `execute` nové třídy, jež je odvozená z připravené třídy `DefaultInternalAction`. Jméno vytvořené třídy odpovídá názvu funkce, kterou může agent zavolat jako běžnou funkci, včetně předání parametrů. Důležitou vlastností těchto funkcí je, že lze získat přístup k představám agenta, který ji zavolá. Dále bude v této podkapitole popsána implementace všech vytvořených funkcí.

Pathfinder

Tato abstraktní třída slouží vyhledání cesty prostředím na základě agentových představ. Pomocí metody `createMap` se z agentových představ vytvoří jednoduchý model rozvržení terénu, jenž agent zná. Dále obsahuje metody pro nastavení počátečního a koncového místa hledané trasy. Jádrem této třídy je metoda `aStar`, která se pokusí najít nejkratší cestu od začátku k cíli. K hledání využívá algoritmus A* (dle literatury [5]), jenž se používá pro nalezení optimální cesty v kladně ohodnoceném grafu. Jako hodnotící funkce je brán počet kroků od začátku plus přímá vzdálenost k cíli, tedy součet absolutních rozdílů vertikálních a horizontálních souřadnic.

Tato funkce využívá objekty vlastní privátní třídy `Node`, které reprezentují dosažená políčka mapy. V tomto objektu se ukládá délka dosavadního postupu a předchůdce, což slouží ke snadné rekonstrukci cesty po dosažení cíle.

Dis, dis_from a path

Tyto tři třídy jsou odvozené ze třídy `PathFinder`, z níž využívají hledání cesty.

První dvě jsou skoro stejné, obě slouží pro zjištění vzdálenosti dvou políček neboli délky nejkratší cesty mezi nimi. Liší se výchozím místem. Zatímco u `dis` je to aktuální pozice agenta, u třídy `dis_from` je to libovolné místo zadané parametrem.

Funkce `path` také hledá nejkratší cestu od pozice agenta k zadanému cíli. Při úspěšném nalezení cesty se ze pomocí metody `back` uloží agentovy do báze znalostí kompletní cesta. Tedy políčka, přes které vede, se směrem dalšího kroku.

End

Tuto funkci volají všichni agenti těsně před ukončením simulace. Slouží k získání statistických údajů, které agenti sbírají během své činnosti. Pomocí této funkce se všechny údaje předají do hlavni statiky.

Kapitola 8

Experimenty

V této kapitole jsou popsány a vyhodnoceny provedené experimenty na vytvořeném simulátoru. Část experimentů, u nichž je provedeno velké množství různých simulací, slouží ke srovnání úspěšnosti jednotlivých režimů umělé inteligence. Vlastnosti těchto režimů jsou pak zjištěny na základě výsledků dalších experimentů.

Na začátku kapitoly budou zmíněny a vysvětleny měřené veličiny a použité typy map prostředí. Zbytek kapitoly bude potom rozdělen na jednotlivé experimenty, které budou detailněji popsány a samostatně vyhodnoceny. Budou zde uvedeny jen pro experiment podstatné výsledky a důležité změřené veličiny. Ostatní údaje jsou uvedeny v příloze E, v níž jsou kompletní výsledky všech experimentů a agregované hodnoty měřených veličin.

Všechny experimenty proběhly na stroji s procesorem *Intel Quad Core i5* o frekvenci *3,3GHz* a *8GB DDR3* pamětí v operačním systému *Arch Linux 3.8.7.* s *Java 1.7.0_21* a *Jason 1.3.6a.*

8.1 Měřené veličiny a použité mapy

Jako měřené veličiny jsem si zvolil několik podstatných údajů, které vypovídají o běhu simulace a jejích výsledcích. Tyto veličiny jsou měřeny a zaznamenány při každém běhu simulace. Následuje seznam všech veličin, jejich popis a případně i zdůvodnění jejich důležitosti. U každé veličiny je uvedena její dále používaná zkratka a jednotka. Jednotky těchto veličin se obvykle vztahují k průměrné hodnotě za deset kol u jednoho agenta, což umožní srovnání jednotlivých simulací bez ohledu na jejich celkovou délku a počet agentů. Měřené veličiny jsou tyto:

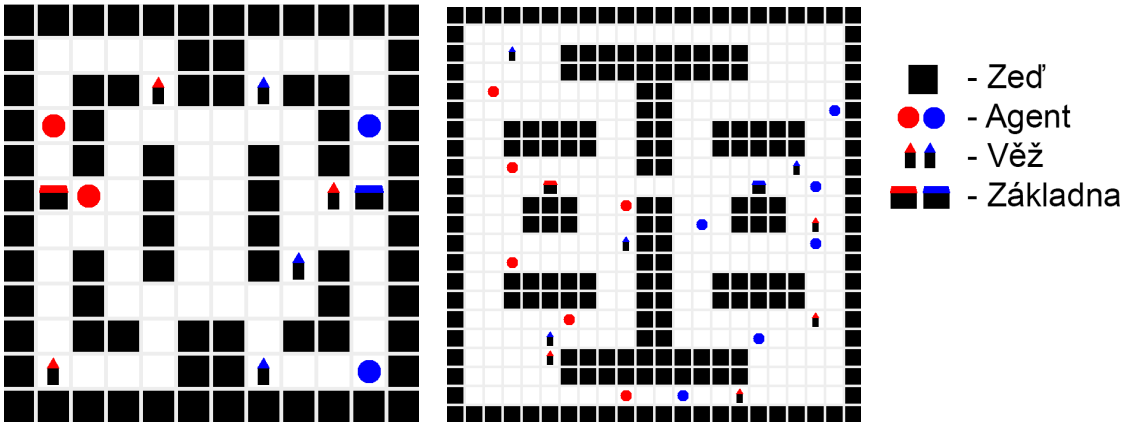
- **TIME** – doba výpočetní fáze deseti kol v *ms*
- **INFOTIME** – doba informační fáze deseti kol v *ms*
- **STEPS** – počet kroků/přesunů za deset kol
- **MSG** – počet zaslaných zpráv za deset kol
- **FCE** – počet volání `path`, `dis` a `dis_from` za deset kol
  Počet volání těchto speciálních funkcí (viz kap. 7.3) je důležitý, protože jsou to funkce výpočetně náročné, jelikož vytváří a prohledávají často velkou část mapy.

- **NZDP** – počet použití pravidla `nzdp` za deset kol
  Jedná se o velmi náročné pravidlo (viz příloha D.2), protože prochází všechna známá políčka a hledá jedno nejvhodnější.
- **RNDS** – celkový počet kol simulace
- **RT** – počet zbylých věží vítězného týmu

Dále v textu jsou uvedeny hodnoty těchto veličin vždy vyjádřeny ve zvolené jednotce, a to jako průměr naměřených hodnot všech agentů v týmu. Pro dosažení objektivních výsledků a snížení vlivu extrémních situací jsou všechny experimenty provedeny vícekrát s různým výchozím rozložením prostředí. Výsledné hodnoty jsou potom spočítány jako průměr naměřených hodnot všech běhů simulace v rámci jednoho experimentu.

Ve všech experimentech, kromě druhého v kapitole 8.3, jsou použity mapy se symetrickým rozvržením terénu a poloh obou základen. Tímto je zrušen vliv rozdílného terénu a žádný tým nezíská značnou výhodu, díky čemuž je lze lépe porovnat. Pro experimenty jsem navrhl tyto typy map, které se liší rozložením terénu:

- **Mini**
  Mapa o velikosti 10x10, která je zobrazena na obrázku 8.1a. Jedná se o nejmenší mapu, vhodnou pro malý počet agentů a věží. Zeď je rozmístěna po malých částech, jež netvoří žádné slepé cesty, a základny u svislých okrajů jsou dobře dostupné.
- **Classic**
  Mapa má velikost 20x20 a je na obrázku 8.1b. Tato mapa je označená jako klasická, protože je střední velikosti a neobsahuje žádné extrémní rozvržení terénu. Zdi jsou sloučeny do nevelikých úseků a mapa neobsahuje žádné slepé cesty. Základny jsou umístěny v její střední části, kde je k nim odkudkoli snadný přístup. Tato mapa je proto použita v největším množství experimentů.

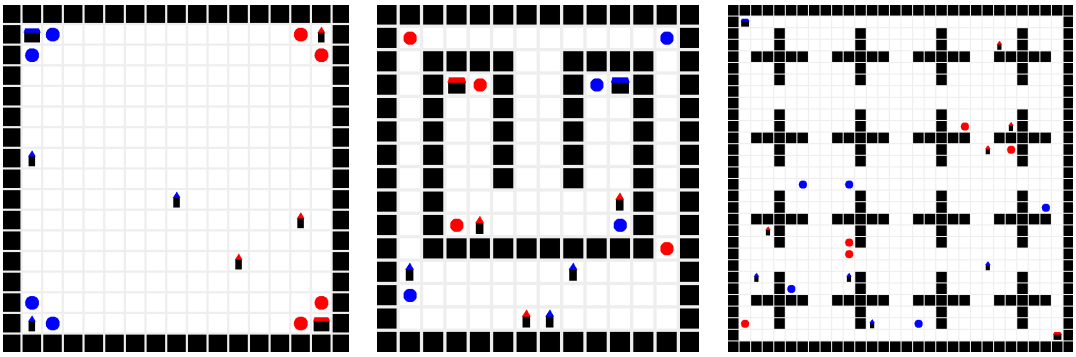


(a) Typ *Mini* (b) Typ *Classic*

Obrázek 8.1: Ukázky map.

- **Free**
  Velikost této mapy je 15x15 a její ukázka je na obrázku 8.2a. Tato extrémní mapa neobsahuje žádnou zeď a slouží tedy k analýze chování agentů v prostředí bez překážek. Základny jsou v protilehlých rozích stejně jako dvě věže.

- **U**
  Tato mapa, která má velikost 13x12, je na obrázku 8.2b. Jedná se o další extrémní mapu. Zeď je spojena do jednoho kusu ve tvaru *U*, čímž rozděluje mapu na částí mezi kterými není snadná cesta. Tato mapa se vyznačuje zejména obtížným přístupem k základnám, k nimž vedou pouze dvě cesty, jež jsou často velmi zdlouhavé.



(a) Typ *Free* (b) Typ *U* (c) Typ *Cross*

Obrázek 8.2: Ukázky map.

- **Cross**
  Tato speciální mapa proměnné velikosti je na obrázku 8.2c. Mapa je vždy složena ze stejných samostatných částí o velikosti 7x7. Jedna tato část obsahuje jeden celý kříž a jeho okolí. Díky tomuto lze vytvářet libovolně velké mapy, které mají pravidelný terén. Základny jsou zde umístěny do protilehlých rohů.

Rozvržení terénu a umístění základen je při použití těchto map u všech experimentů zachováno. U některých simulací jsou také zachovány zobrazené symetrické výchozí pozice vybraných agentů a věží, ke kterým jsou další přidány náhodně. U ostatních simulací jsou všichni agenti a věže rozmístěny náhodně.

8.2 Experiment 1 – vítězství více spolupracující inteligence

Tento první experiment byl stěžejní k porovnání celkové činnosti jednotlivých režimů inteligence. Jeho cílem bylo zjistit, zda více spolupracující tým zvítězí v nejrůznějších prostředích při rovnocenných podmínkách. Simulace v tomto experimentu byly provedeny pro všechny kombinace týmů s různými režimy inteligence, tedy UI1 proti UI2, UI2 proti UI3 a UI1 proti UI3.

Experiment jsem rozdělil do dvou částí. V první části jsem se snažil o rovnocenné podmínky obou týmů, které by vedli k přesnějšímu vzájemnému srovnání. Oba týmy měli zrcadlově stejné rozmístění agentů a některých věží. Zbylé dvě věže byly pro každý tým v každém běhu simulace do těchto map náhodně generovány.

Pro tuto část jsem vybral tři různé typy map s následujícím počtem objektů v týmu:

- *Classic*: šest agentů a čtyři věže
- *Free*: čtyři agenti a tři věže
- *U*: čtyři agenti a tři věže

V rámci této části experimentu bylo provedeno deset opakování pro každou dvojici týmů na každé mapě, celkem tedy devadesát simulací. Souhrnné výsledky pro všechny mapy jsou uvedeny v tabulce 8.1, která ukazuje dominanci UI3, ale srovnatelné výsledky UI1 a UI2. Časté výhry režimu bez spolupráce v této části experimentu byly převážně na mapě typu *Free*. Tyto výhry byly zapříčiněny zejména náhodným rozmístěním věží — výhodným pro jeden tým — při malém počtu opakování.

Režim inteligence	**UI1**	**UI2**	**UI3**
UI1	-	15 (17)	6 (6)
UI2	15 (22)	-	8 (8)
UI3	24 (40)	22 (35)	-

Tabulka 8.1: Počet výher a v závorce celkový počet zbylých věží jednotlivých režimů, 1. experiment 1. část.

Ve druhé části experimentu bylo rozmístění všech věží a agentů voleno náhodně. Pro tuto část jsem si zvolil pouze mapu typu *Classic*, kde bylo při každém běhu do týmu náhodně vygenerováno šest věží a čtyři agenti. Pro všechny dvojice týmů bylo provedeno dvacet opakování. Výsledky těchto simulací jsou shrnuty v tabulce 8.2, ze které je patrná převaha více spolupracující inteligence i při zcela náhodném počátečním rozmístěním všech objektů.

Režim inteligence	**UI1**	**UI2**	**UI3**
UI1	-	6 (10)	5 (8)
UI2	14 (24)	-	6 (10)
UI3	15 (30)	14 (27)	-

Tabulka 8.2: Počet výher a v závorce celkový počet zbylých věží jednotlivých režimů, 1. experiment 2. část.

Z tabulky 8.3, kde je sečten celkový počet vítězství a zbylé věže z obou částí experimentu, vyplývá jasná převaha inteligence, ve které agenti více spolupracují, a to v nejrůznějších typech prostředí.

Režim inteligence	UI1	UI2	UI3
Celkový počet výher (RT)	32 (41)	43 (64)	75 (132)

Tabulka 8.3: Celkový počet výher a v závorce celkový počet zbylých věží jednotlivých režimů, 1. experiment.

Protože je v tomto experimentu provedeno nejvíce simulací pro všechny režimy agentů, zvolil jsem si jej pro porovnání všech naměřených veličin mezi úrovněmi inteligence. V tabulce 8.4 jsou uvedeny průměrné naměřené hodnoty ze všech provedených simulací v rámci tohoto experimentu, které jsou rozděleny dle režimů inteligence.

Důležitým zjištěním tohoto experimentu je, že agenti, kteří nespolupracují, potřebují k rozhodnutí o další akci v průměru o polovinu víc času něž ti, co spolupracují, a to i přes to, že je jejich rozhodovací proces složitější. Je to způsobeno tím, že spolupracující agenti nemusejí volit další akci v každém kole, ale často mají další činnost naplánovanou od jiného agenta nebo plní funkci podpory, u které se nemusejí příliš rozhodovat.

Režim inteligence	TIME	INFOTIME	STEPS	MSG	FCE	NZDP
UI1	67	0	9	0	23	1
UI2	42	2	8	44	18	< 1
UI3	45	3	8	50	77	$<< 1$

Tabulka 8.4: Průměrné hodnoty měřených veličin u jednotlivých režimů (TIME a INFOTIME – *ms*, ostatní – četnost za deset kol), 1. experiment.

Počet zpráv a doba informační části je u režimu komplexní spolupráce mírně větší než u režimu částečné spolupráce, což odpovídá většímu množství předávaných informací.

V režimu komplexní spolupráce volají agenti speciální funkce, které hledají cestu nebo vzdálenost, třikrát častěji něž ostatní agenti. Tito agenti při svém rozhodování totiž také uvažují a počítají vzdálenost ostatních agentů. Přesto mají srovnatelnou dobu rozhodování jako agenti s částečnou spoluprací, a to proto, že si plánují navzájem a svým delším rozhodováním šetří čas kolegů.

Agenti v režimech, které spolupracují, udělají průměrně méně kroků, protože sdílejí rozložení mapy a všichni agenti nemusejí projít každé políčko, aby ho objevili, takže nedělají zbytečné kroky a věnují se podstatnější činnosti.

8.3 Experiment 2 – vítězství na náhodné mapě

Tento experiment byl, stejně jako ten předchozí, zaměřen na srovnání úspěšnosti všech režimů inteligence, ale ve zcela náhodném prostředí. Simulace probíhaly na náhodné čtvercové mapě se stranou o velikosti 15–30 políček. Na této mapě byl v každém běhu náhodně rozmístěn terén, základny, všichni agenti a věže. Oba týmy měly vždy stejný počet agentů (3–6) a věží (3–6).

V tomto experimentu bylo také pro každou dvojici týmů provedeno deset opakování. Dosažené výsledky jsou zaznamenány v tabulce 8.5.

Režim inteligence	UI1	UI2	UI3
UI1	-	3 (3)	2 (5)
UI2	7 (12)	-	4 (7)
UI3	8 (17)	6 (14)	-

Tabulka 8.5: Počet výher a v závorce celkový počet zbylých věží jednotlivých režimů, 2. experiment — náhodné mapy.

Z výsledků tohoto experimentu vyplývá převaha více spolupracujícího týmu, i když ne tak zřejmá jako v předchozím experimentu. Toto dokazuje, že náhodnost prostředí nemá velký vliv na činnost jednotlivých režimů a více spolupracující tým porazí méně spolupracující i při nestejných podmínkách.

8.4 Experiment 3 – vliv velikosti mapy

Tento experiment měl zjistit, jaký má velikost mapy vliv na měřené veličiny, tedy jak ovlivňuje velikost mapy činnost agenta. Proti sobě soupeřili jen režimy UI1 a UI3, což bylo dostačující, protože měřené hodnoty režimu UI2 se nacházely mezi nimi. Simulace byly

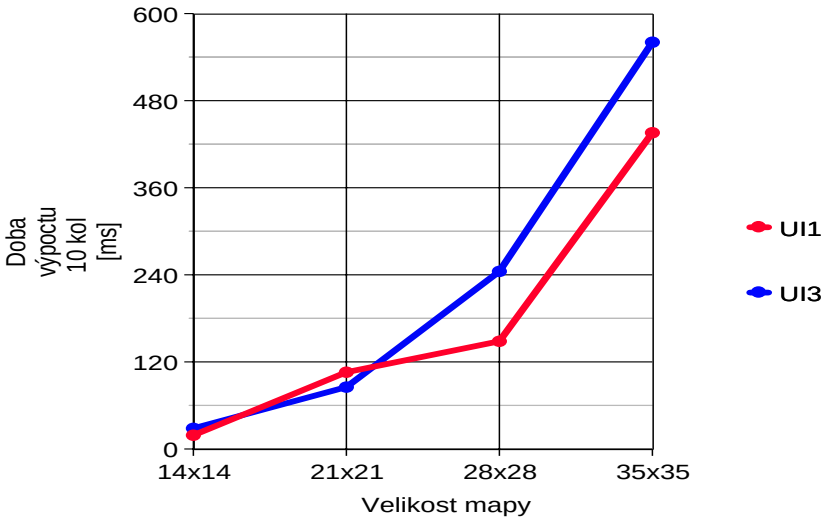
provedeny na mapě typu *Cross*, jež je vhodná pro svou proměnlivou velikost. Byly použity mapy se stranou o velikosti 14, 21, 28 a 35 políček. Pro každou velikost uskutečnily čtyři souboje, v nichž bylo pro každý tým náhodně rozmístěno pět agentů a čtyři věže. Naměřené hodnoty jsou v tabulce 8.6.

Velikost mapy	**TIME**		**INFOT.**		**STEPS**		**MSG**		**FCE**		**RNDS**
	UI1	UI3	UI1	UI3	UI1	UI3	UI1	UI3	UI1	UI3	
14x14	19	28	0	4	9	8	0	49	19	124	374
21x21	106	86	0	4	9	9	0	48	20	54	545
28x28	149	244	0	4	9	9	0	47	24	74	810
35x35	435	561	0	5	10	9	0	48	25	51	965

Tabulka 8.6: Průměrné hodnoty podstatných veličin (TIME a INFOTIME – *ms*, ostatní – četnost za deset kol), 3. experiment — různé velikosti mapy.

Podle zjištěných údajů má velikost mapy zásadní vliv na dobu, kterou agent potřebuje pro výpočet další akce. Tato doba s velikostí mapy exponenciálně roste, což je názorné na obrázku 8.3. Na tomto obrázku je zvláštní, že u mapy o velikosti 21x21 políček je doba výpočtu další akce agentů UI1 delší než UI3, což je způsobeno malým počtem opakování simulace. Z těchto výsledků plyne, že jsou na velkých mapách všechny úrovně agentů nepoužitelné, protože by se agenti rozhodovali příliš dlouho. Doba rozhodování je závislá především na speciálních funkcích. Počet jejich volání zůstává srovnatelný (mimo UI3 na nejmenší mapě, kdy průměrnou hodnotu zkresluje nízký počet kol simulace), ale prohledávají větší část prostředí, k čemuž potřebují delší dobu.

Na jiné veličiny velikost mapy vliv nemá. Průměrný počet kroků agentů a kol simulace je vyšší, protože jsou budovy více vzdáleny.



Obrázek 8.3: Závislost doby výpočtu další akce na velikosti mapy.

8.5 Experiment 4 – vliv počtu agentů

Experiment podobný předchozímu, který měl zjistit vliv počtu agentů na měřené veličiny. Opět proti sobě soupeřily pouze režimy UI1 a UI3, ale na mapě typu *Classic*, na které byly vždy náhodně rozmístěny čtyři věže do týmu. Oba týmy měli stejný počet agentů, a to postupně 3,6,9 a 12, se kterými se vždy provedli čtyři opakování. Výsledky experimentu jsou v tabulce 8.7.

Počet agentů	**TIME**		**INFOT.**		**STEPS**		**MSG**		**FCE**		**RNDS**
	UI1	UI3	UI1	UI3	UI1	UI3	UI1	UI3	UI1	UI3	
3 vs. 3	82	44	0	3	9	8	0	24	21	31	835
6 vs. 6	120	78	0	4	9	8	0	63	27	164	320
9 vs. 9	186	75	0	4	9	8	0	101	38	282	305
12 vs. 12	71	115	0	5	9	9	0	146	31	169	164

Tabulka 8.7: Průměrné hodnoty podstatných veličin (TIME a INFOTIME – *ms*, ostatní – četnost za deset kol), 4. experiment — různý počet agentů.

Z experimentu plyne, že množství agentů neovlivní jejich činnost tak jako velikost mapy. Počet agentů v týmu ovlivní pouze spolupracující režimy, protože při rozhodování uvažují nad svými kolegy, k čemuž většinou používají speciální funkce. Počet volání těchto funkcí tedy exponenciálně roste s počtem kolegů (mimo UI3 na poslední mapě — opět málo kol simulace), ale jejich časté použití na této malé mapě jen nepatrně prodlouží dobu rozhodování.

Počet zaslaných zpráv a doba informační fáze roste s počtem agentů lineárně, protože všem agentům v týmu zasílají převážně stejné zprávy.

8.6 Experiment 5 – přesila

V tomto experimentu se porovnávala úspěšnost týmu spolupracujících agentů proti přesile týmu nespolupracujících. Cílem tohoto experimentu bylo zjistit, jaký počet agentů bez spolupráce by překonal menší počet agentů s komplexní spoluprací. Proto byly použity režimy UI1 a UI3. Simulace probíhaly na mapě typu *Classic* se čtyřmi věžemi na tým. V týmu komplexní spolupráce bylo vždy pět agentů. V týmu s UI1 bylo 6–10 protivníků, s nimiž se provedlo sedm opakování. Výsledky soupeření jsou v tabulce 8.8.

Počet agentů	**Počet výher (RT)**	
	UI1	UI3
6 vs. 5	4 (7)	3 (5)
7 vs. 5	6 (11)	1 (1)
8 vs. 5	7 (12)	0 (0)
9 vs. 5	7 (15)	0 (0)
10 vs. 5	7 (16)	0 (0)

Tabulka 8.8: Počet výher a v závorce celkový počet zbylých věží při vzájemných soubojích, 5. experiment — přesila.

Z výsledků vyplývá, že při přesile o jednoho agenta má početnější tým mírnou převahu, ale úspěšnost obou týmů je vyrovnaná a rozhoduje především rozvržení prostředí. Při přesile

o dva agenty získali agenti UI1 většinu vítězství, ale tým režimu komplexní spolupráce je schopen zvítězit. Při přesile o tři a více agentů má početnější tým absolutní převahu, protože má o polovinu víc agentů než UI3, který není schopen vyhrát

8.7 Experiment 6 – nízký počet agentů

Tento experiment měl porovnat úspěšnost režimů inteligence, které měly nízký počet agentů v týmu. Cílem bylo zjistit, v jakém nejnižším počtu dokážou agenti výhodně spolupracovat a jestli se potom jejich spolupráce nestane přítěží. Použita byla nejmenší mapa typu *Mini* se třemi věžemi a 1–3 agenty na tým. V rámci tohoto experimentu se provedlo s každým rozložením sedm opakování, jejichž výsledky jsou zapsány v tabulce 8.9.

Počet agentů	**Počet výher (RT)**	
	UI1	UI3
1 vs. 1	3 (3)	4 (6)
2 vs. 2	1 (1)	6 (9)
3 vs. 3	1 (1)	6 (10)

Tabulka 8.9: Počet výher a v závorce celkový počet zbylých věží při vzájemných soubojích, 6. experiment — málo agentů.

Tento experiment ukázal, že jsou dokonce i dva agenti režimu komplexní spolupráce schopni výhodně spolupracovat a zvítězit ve většině simulací. V souboji jeden na jednoho jsou režimy vyrovnány, takže spolupracující agent není v nevýhodě, ale rozhoduje rozvržení prostředí.

Kapitola 9

Závěr

Cílem této práce bylo navrhnout a vytvořit multiagentní systém včetně modelu prostředí a úlohy se zaměřením na spolupráci. V tomto systému mělo být realizováno několik různých režimů inteligence agentů, které mají být otestovány, vyhodnoceny a porovnány.

Abych mohl tuto problematiku řešit, musel jsem nastudovat teorii multiagentních systémů a způsoby jejich realizace. Seznámil jsem se s platformou Jason a zejména s jeho rozšířenou verzí jazyka AgentSpeak(L), který se používá k tvorbě BDI agentů. Dále jsem nastudoval problematiku modelování a simulace včetně tvorby a vyhodnocování experimentů.

Výsledkem této práce je simulátor multiagentního systému. V tomto simulátoru je implementováno navržené prostředí a úloha. Prostředí je 2D matice složena z jednotlivých políček, n nichž může stát zeď nebo budova. Po této mapě se pohybují agenti, kteří jsou rozděleni do dvou týmů, jež proti sobě soupeří. Jejich soupeření spočívá ve vzájemném ničení a opravování věží za použití nástrojů, které se zpracovávají a uchovávají v základně.

Simulátor také obsahuje tři navržené režimy inteligence agentů, jež se liší mírou vzájemné spolupráce. Agenti všech těchto režimů se snaží co nejlépe úlohu splnit, ale odlišnými technikami. V prvním režimu bez spolupráce agenti nijak nekomunikují a nespolupracují ani o sobě navzájem neuvažují. V druhém režimu mírné spolupráce si agenti zasílají některé informace o prostředí a jejich vzájemná spolupráce spočívá v tom, že v případě potřeby žádají o podporu své kolegy. V posledním režimu komplexní spolupráce si agenti předávají veškeré údaje o stavu prostředí a vzájemně si dopředu plánují svoji další činnost.

V rámci implementace simulátoru, jenž je i s prostředím realizován pod frameworkem Jason v jazyce Java, jsem vytvořil vizualizaci, která zobrazuje průběh simulace, tedy aktuální stavy prostředí při jejím běhu. Tato vizualizace má dvě verze: schématickou a podrobnou.

Pro demonstraci a analýzu chování jednotlivých režimů inteligence jsem vytvořil sadu experimentů, v rámci nichž jsem provedl a vyhodnotil přes 250 různých simulací. Na základě výsledků těchto experimentů jsem zjistil, že míra spolupráce agentů je klíčovým faktorem k jejich vítězství, a to proto, že ve většině souměrných i náhodných prostředích zvítězila inteligence, ve které agenti více spolupracovali. Dále experimenty prokázaly, že agenti jsou schopni výhodně spolupracovat, i pokud jsou v týmu pouze dva. V souboji jeden na jednoho jsou potom režimy vyrovnané. Z dalšího experimentu plyne, že jsou agenti v režimu komplexní spolupráce srovnatelní s 20% přesilou agentů v režimu bez spolupráce. Větší přesila už v drtivé většině případů vítězí.

Experimenty také ukázaly, že má velikost mapy zásadní vliv na dobu, kterou agenti všech režimů potřebují k výběru další akce. Tato doba s velikostí mapy exponenciálně roste, proto jsou agenti efektivně použitelní jen na nevelikých mapách (do cca 60x60 políček,

dle použitého stroje), jinak se rozhodují příliš dlouho. Naopak větší počet agentů v týmu prodlouží dobu rozhodnutí jen nepatrně, a to jen u agentů, jenž spolupracují.

Vytvořený simulátor lze také použít i pro jiné režimy inteligence, jež by mohly využít prostředí a řešit zadanou úlohu. Upravit stávající režim inteligence nebo vytvořit nový lze provést jednoduše, a to změnou jediného souboru s jádrem rozhodovacího procesu inteligence. Nový režim by mohl využít vytvořených pravidel, plánů k přesunu agenta a plánů pro vykonaní akce, které by řídily činnosti agentů a zajistily požadovanou funkčnost na vysoké úrovni abstrakce.

Další vývoj tohoto systému bych zaměřil na zlepšení činnosti agentů v režimu komplexní spolupráce, jenž se ukázal jako nejlepší. Tato inteligence by se dále měla testovat v nejrůznějších situacích, kterým by se měla postupně přizpůsobit a vyvíjet se tak, aby dosáhla maximální racionality v co nejvíce odlišných případech.

Literatura

[1] Bordini, R. H.; Hübner, J. F.: BDI agent programming in agentspeak using Jason. In *Proceedings of the 6th international conference on Computational Logic in Multi-Agent Systems*, CLIMA'05, Berlin, Heidelberg: Springer-Verlag, 2006, ISBN 3-540-33996-5, 978-3-540-33996-0, s. 143–164.

[2] Bordini, R. H.; Wooldridge, M.; Hübner, J. F.: *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007, ISBN 0470029005.

[3] Fishwick, P. A.: *Simulation Model Design and Execution: Building Digital Worlds*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995, ISBN 0130986097.

[4] Rao, A. S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, MAAMAW '96, Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, ISBN 3-540-60852-4, s. 42–55.

[5] Russell, S. J.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson Education, druhé vydání, 2003, ISBN 0137903952.

[6] Weiss, G. (editor): *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, MA, USA: MIT Press, 1999, ISBN 0-262-23203-0.

[7] Wooldridge, M.; Jennings, N. R.: Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, ročník 10, č. 2, 1995: s. 115–152.

[8] Woolridge, M.: *Introduction to Multiagent Systems*. New York, NY, USA: John Wiley & Sons, Inc., 2002, ISBN 047149691X.

[9] Jason | a Java-based interpreter for an extended version of AgentSpeak. [online, navštívené 13.4.2013].
URL http://jason.sourceforge.net/wp/

Příloha A

Obsah CD

Přiložené CD obsahuje tyto adresáře:

- **tex** – textová část práce ve formátu `pdf`, včetně zdrojových kódů v LaTeXu
- **src** – zdrojové kódy prostředí a funkcí
- **ui** – zdrojové kódy umělé inteligence agentů
- **map** – použité mapy prostředí
- **exp** – kompletní výsledky všech provedených experimentů
- **pole** – výsledná spustitelná aplikace

Příloha B

Použití aplikace

Výsledný simulátor se spouští příkazem `java -jar Pole.jar`. Simulace se řídí zvolenými parametry a mapou, jejichž formát a význam je popsán níže.

Parametry

Povinné jsou parametry, které udávají režim agentů a plně určují výchozí stav mapy.

- `-r N` – režim agentů červeného týmu, kde N je 1–3
- `-b N` – režim agentů modrého týmu
- `-map FILE` – soubor s mapou
- `-rmap N` – vygeneruje se náhodná čtvercová mapa o straně N bez věží a agentů
- `-rag N` – přidá N agentů do každého týmu na náhodné pozice
- `-rtow N` – přidá N věží pro každý tým na náhodné pozice
- `-gui/-gui2` – průběh simulace je vizualizován s detaily/schématicky
- `-o FILE` – soubor, do kterého se uloží výsledky a statistika simulace

Mapa

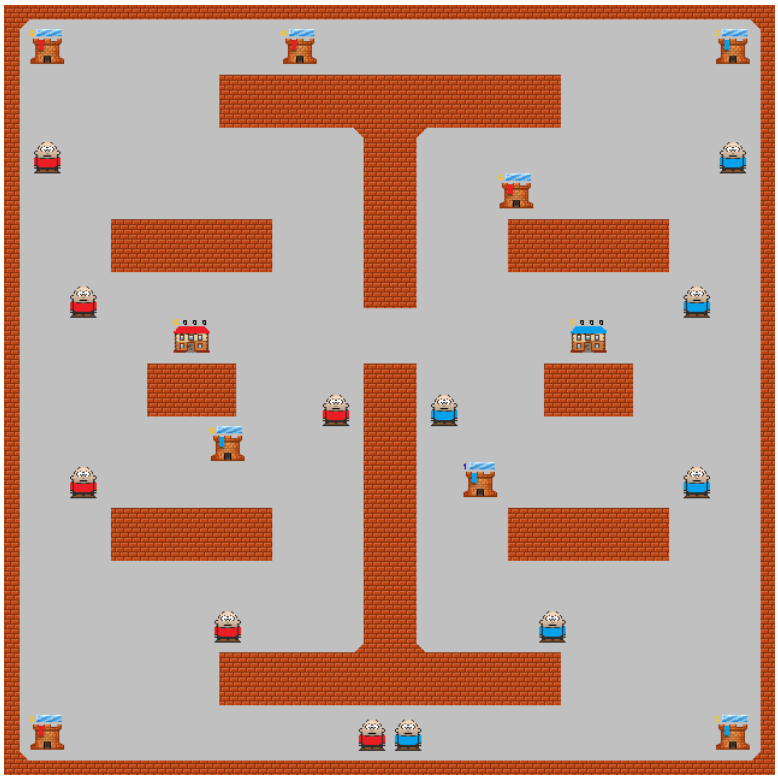
Výchozí stav prostředí je dán mapou, kterou je možné načíst ze souboru. Tento soubor obsahuje její textovou reprezentaci, v níž každý znak určuje výchozí obsah právě jednoho políčka. Mapa musí být zadána úplná, žádné políčko tedy nelze vynechat. Pro přidání dalších náhodných objektů slouží parametry (viz výše). Mapa může obsahovat pouze následující znaky:

- `x, X, w, W` – zeď
- `B` – základna červeného týmu
- `T` – věž červeného týmu
- `A` – agent červeného týmu
- `o, 0` – prázdné volné políčko
- `b` – základna modrého týmu
- `t` – věž modrého týmu
- `a` – agent modrého týmu

Příklad mapy typu *Classic*:

```
TooooooTooooooooooot
ooooooxxxxxxxxxooooo
ooooooxxxxxxxxxooooo
Aooooooooxxoooooooooa
ooooooooooxxooTooooooo
ooxxxxxooxxooxxxxxoo
ooxxxxxooxxooxxxxxoo
oAooooooooxxoooooooao
ooooBooooooooooboooo
oooxxxooooxxoooxxxooo
oooxxxooAxxaooxxxooo
oooootoooxxoooooooo
oAooooooooxxotooooao
ooxxxxxooxxooxxxxxoo
ooxxxxxooxxooxxxxxoo
ooooooooooxxoooooooo
ooooooAoooxxoooaooooo
ooooooxxxxxxxxxooooo
ooooooxxxxxxxxxooooo
TooooooooAaooooooooot
```

Grafická podoba této mapy je na obrázku B.1.



Obrázek B.1: Mapa typu *Classic*.

Příloha C

Akce agenta

Tato příloha obsahuje seznam všech možných akcí agenta. U každé akce je její krátký popis, podmínka, kdy může být akce provedena, a změna prostředí po provedení akce. Některé akce jsou navíc podmíněny poměrem počtu agentů v okolí, více v kapitole 5.6. Všechny akce trvají jedno kolo.

- **Vzít nástroj:** Agent si vezme cihlu nebo zbraň ze své základny.
  Podmínka: Agent stojí na svojí základně. V základně je alespoň jeden nástroj vybraného druhu.
  Po provedení: Počet nástrojů v základně se o jeden sníží. Agent vlastní vybraný nástroj.

- **Zaútočit:** Agent zaútočí na soupeřovu věž.
  Podmínka: Agent stojí na soupeřově nezničené věži a má u sebe zbraň.
  Po provedení: Agent nemá žádný nástroj. Snížena hodnota života věže.

- **Opravit:** Agent opraví svoji věž.
  Podmínka: Agent stojí na své věži a má u sebe cihlu.
  Po provedení: Agent nemá žádný nástroj. Zvýšena hodnota života věže.

- **Pohnout se:** Agent si zvolí jeden ze směrů: nahoru, dolů, doprava, doleva. Tímto směrem udělá krok na sousední políčko.
  Podmínka: Cílové políčko není zeď.
  Po provedení: Změní se poloha agenta ve zvoleném směru.

- **Vytěžit suť:** Agent vezme jeden kus suti ze zničené věže.
  Podmínka: Agent stojí na zničené věži, kde je alespoň jeden kus suti.
  Po provedení: Agent má u sebe kus suti. Množství sutí o jeden kus sníženo.

- **Uložit nástroj:** Agent uloží nástroj, který vlastní, do své základny.
  Podmínka: Agent vlastní nástroj a stojí na své základně.
  Po provedení: Agent nemá žádný nástroj. Množství příslušných nástrojů na základně se o jeden kus zvýšilo.

- **Vykrást:** Agent ukradne nástroj ze soupeřovi základny.
  Podmínka: Agent stojí na soupeřově základně, ve které je alespoň jeden vybraný nástroj.
  Po provedení: Agent vlastní vybraný nástroj. Ze skladu základny ubyl jeden nástroj.

- **Vyrobit cihlu:** Agent vytvoří na své základně ze suti nové cihlu.
  Podmínka: Agent stojí na své základně, ve které je alespoň jeden kousek suti.
  Po provedení: Na základně se snížil počet kusů suti o jeden a zvýšil počet cihel.

- **Vyrobit zbraň:** Agent vytvoří na své základně novou zbraň.
  Podmínka: Agent stojí na své základně.
  Po provedení: Na základně se zvýšil počet zbraní.

- **Prázdná akce:** Agent neprovede žádnou akci.
  Podmínka: Žádná.
  Po provedení: Nic se nezmění.

Příloha D

Doplňující informace k implementaci

V této příloze jsou popsány doplňující informace k implementaci prvních dvou modulů. Pomocné třídy z modelu prostředí a sdílené soubory inteligence agentů.

D.1 Model prostředí

Tato sekce obsahuje popis pomocných tříd v modelu prostředí. Rozšiřuje kapitolu 7.1 — první modul implementace.

Level

Pomocná třída, jejíž instance reprezentuje počáteční rozložení prvků prostředí — mapu. Mapu načte buď ze zadaného souboru, nebo podle zvolených parametrů náhodně vygeneruje. Tuto mapu uchovává v textové podobě. Každý znak (viz příloha B) určuje obsah jednoho políčka, na němž může být tedy pouze jeden objekt. Tato třída také ověřuje správnost mapy, která musí mít stejnou délku všech řádků. Dále musí obsahovat právě jednu základnu a nejméně jednu věž a agenta od každého týmu.

Stats a StatsAgent

Tyto dvě třídy slouží k tvorbě statistik o běhu simulace. Objekt třídy `StatsAgent` uchovává vybrané statistické údaje o činnosti jednoho agenta. Pro všechny agenty jsou tyto objekty sdruženy do seznamu v objektu třídy `Stats`, pomocí kterého se k nim přistupuje. Tento objekt po skončení simulace všechny zjištěné údaje projde a vytvoří z nich souhrnnou statistiku jednotlivých týmů.

PObject, Agent, Base a Tower

Třída `PObject` slouží jako rodičovská třída pro všechny objekty v modelu prostředí, které nejsou triviální a mají vnitřní stav. Tento rodič uchovává společné vlastnost všech těchto objektů, jako jsou poloha a tým. Také obsahuje u agenta nevyužitý stav blokování budovy a metodu na jeho výpočet, která využívá metod modelu.

Třída `Agent`, jejíž instance reprezentuje jednoho agenta, navíc obsahuje jméno a id agenta. Dále přibyla informace o agentem drženém nástroji a metody pro jeho přidávání a odebírání.

Základnu představuje objekt třídy `Base`, jež je rozšířená o počty jednotlivých nástrojů, které jsou v základně, včetně metod pro jejich manipulaci.

Věž je reprezentována pomocí třídy `Tower`, která navíc uchovává stav života věže, případně množství suti, jež je vyjádřeno zápornou hodnotou života. Metody pro snižování a zvyšování života hlídají meze, aby nedošlo k opravení zničené věže nebo aby hodnota života nepřekročila výchozí maximální hranici.

D.2 Inteligence

V této části budou popsány čtyři soubory, jenž při rozhodování využívají agenti všech režimů inteligence. Každý tento soubor obsahuje související mechanismy a prvky činnosti agentů. Doplňuje kapitolu 7.2 — druhý modul implementace.

Action

V tomto souboru jsou pomocné plány po vykonaní akce v prostředí. Samotné volání akce na model prostředí je obaleno plány, jež zaznamenávají čas začátku rozhodování a jeho ukončení. Navíc se okamžitě po vykonání činnosti spouští plány, které zajišťují postupné dvoufázové řízení činnosti agentů, jak bylo popsáno výše. Také se zde ověřuje, zda není vykonaní akce zablokováno. V tom případě vykoná prázdnou akci a původní akci vyzkouší v dalším kole. Toto opakuje maximálně desetkrát. Je-li akce stále blokována, ruší celý plán. Celá tato činnost je zapouzdřena pomocí plánu `action`. Všechny akce agenta se volají skrze tento plán, který zajistí jejich korektní průběh.

Dále tento soubor obsahuje mechanismus na zrušení plánu na základě přijaté zprávy. Přijaté zprávy se zpracovávají mimo rozhodovací cyklus. Kdyby byl plán zrušen mimo něj, poruší se mechanismus, jenž pozastavuje činnost agenta, čímž by se narušila postupná činnost agentů. Proto se v takovém případě nastaví příznak `fail`, který je testován na začátku výpočetní cyklu, kdy se může plán zrušit korektně.

Rules

Tento soubor obsahuje pouze pravidla, která usnadňují práci s představami. Pravidla se dají rozdělit do následujících skupin:

- Výpočet vzdálenosti
- Hledání nejbližších objektů (ruiny, poškozená věž, kolega, atd.)
- Je/není něco v okolí něčeho
- Dotazy: odblokuji budovu, opustí/přijde do okolí, můžu provést akci

Velice důležitým pravidlem, jež nepatří do žádné skupiny je `nzdp`. Toto pravidlo je využíváno při hledání cesty přes neznámou oblast dle návrhu v kapitole 6.1. Pomocí tohoto pravidla najde agent políčko, které je dostupné (zná k němu cestu), sousedí z neznámým políčkem a je nejblíže od něj a pozice cíle. Při velkém počtu známých políček je toto pravidlo velmi náročné, protože prochází všechny políčka a hledá nejvhodnější.

Stats

Tento soubor obsahuje plány pro sběr statistik. Důležitý je plán, jenž se spouští před ukončením činnosti agenta, ve kterém se ukládají celkové statistiky o běhu agenta. Tento plán je vykonáván agenty postupně, funguje na známém principu předávání štafety. Postupné vykonání slouží k tomu, aby všichni agenti uložili svoji statistku před ukončením simulátoru, což provede až poslední agent.

Move

Tento soubor je zaměřen na pohyb agenta. Obsahuje plány, které umožňují ovládat pohyb agenta na vysoké úrovni bez znalostí jednotlivých kroků. Těmto plánům stačí předat souřadnice cílového místa a samy zajistí kompletní přesun agenta, včetně vyhledání vhodné cesty.

Všechny tyto plány používají k pohybu plán `go`, který zjistí vhodný směr, a pomocí plánu `krok` udělá tímto směrem jeden krok. V tomto plánu se navíc ověřuje jestli může krok udělat. Nesmí odblokovat svoji budovu nebo opustit okolí budovy, jíž má podpořit. V takových případech provede prázdnou akci a krok se zkusí udělat v příštím kole.

Pro získání směru dalšího kroku se využívá plánu `dalsiKrok`, jenž vychází z návrhu 6.1. Tento plán používá k vyhledání cesty speciální funkci `path` (viz kap. 7.3). Celou nalezenou cestu si uloží do báze znalostí, odkud potom získává směr dalších kroků, které vedou k cíli.

Příloha E

Výsledky experimentů

Zde jsou zobrazeny veškeré souhrnné výsledky z provedených experimentů. Význam měřených veličin a detail použitých map viz kapitola 8.1.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Náhodná čtvercová o straně 15–30 polí	10	UI1	3	3	1103	199	0	9	0	15	2
		UI2	7	12		161	2	9	35	17	0
		UI1	2	5	1414	268	0	9	0	22	1
		UI3	8	17		188	4	9	33	61	0
		UI2	4	7	1180	133	2	8	35	18	0
		UI3	6	14		128	3	8	38	49	0

Tabulka E.1: Souhrnné výsledky 2. experimentu.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Cross 14x14	4	UI1	0	0	374	19	0	9	0	19	1
		UI3	4	9		28	4	8	49	124	0
Cross 21x21	4	UI1	1	2	545	106	0	9	0	20	2
		UI3	3	6		86	4	9	48	54	1
Cross 28x28	4	UI1	1	1	810	149	0	9	0	24	1
		UI3	3	4		244	4	9	47	74	1
Cross 35x35	4	UI1	0	0	965	435	0	10	0	25	2
		UI3	4	12		561	5	9	48	51	1

Tabulka E.2: Souhrnné výsledky 3. experimentu.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Classic	10	UI1	3	3	458	69	0	9	0	21	1
		UI2	7	9		49	2	9	54	17	1
		UI1	0	0	444	62	0	9	0	21	1
		UI3	10	20		71	4	8	62	123	1
		UI2	2	2	484	68	3	8	54	20	1
		UI3	8	15		57	3	8	63	70	1
Free	10	UI1	7	9	339	37	0	9	0	19	1
		UI2	3	5		23	2	9	33	15	0
		UI1	5	5	352	29	0	9	0	19	1
		UI3	5	6		22	3	8	38	47	0
		UI2	4	4	395	17	2	8	33	13	0
		UI3	6	10		18	3	8	37	39	0
U	10	UI1	5	5	723	36	0	9	0	19	1
		UI2	5	8		23	2	9	32	17	0
		UI1	1	1	641	42	0	9	0	26	1
		UI3	9	14		28	3	9	33	38	0
		UI2	2	2	656	23	2	8	32	17	0
		UI3	8	10		26	3	9	33	35	0
Classic zcela náhodné roz-místění objektů	20	UI1	6	10	320	141	0	9	0	30	2
		UI2	14	24		71	2	8	56	22	1
		UI1	5	8	367	120	0	8	0	30	2
		UI3	15	30		66	4	8	64	130	0
		UI2	6	10	328	64	3	8	56	24	1
		UI3	14	27		73	3	8	67	138	0

Tabulka E.3: Souhrnné výsledky 1. experimentu.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Classic 3 vs. 3	4	UI1	3	4	835	82	0	9	0	21	1
		UI3	1	2		44	3	8	24	31	0
Classic 6 vs. 6	4	UI1	0	0	320	120	0	9	0	27	2
		UI3	4	7		78	4	8	63	164	0
Classic 9 vs. 9	4	UI1	2	2	305	186	0	9	0	38	3
		UI3	2	2		75	4	8	101	282	0
Classic 9 vs. 9	4	UI1	1	1	164	71	0	9	0	31	3
		UI3	3	7		115	5	9	146	169	1

Tabulka E.4: Souhrnné výsledky 4. experimentu.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Classic 6 vs. 5	7	UI1	4	7	485	65	0	9	0	22	1
		UI3	3	5		63	3	9	49	76	0
Classic 7 vs. 5	7	UI1	6	11	524	133	0	9	0	22	2
		UI3	1	1		58	3	9	48	54	0
Classic 8 vs. 5	7	UI1	7	12	410	130	0	9	0	25	2
		UI3	0	0		73	4	8	51	82	1
Classic 9 vs. 5	7	UI1	7	15	352	110	0	9	0	25	2
		UI3	0	0		61	4	8	50	85	1
Classic 10 vs. 5	7	UI1	1	1	320	109	0	9	0	25	2
		UI3	0	0		81	4	8	50	77	0

Tabulka E.5: Souhrnné výsledky 5. experimentu.

Mapa	Opak.	Režim	Výhry	RT	RNDS	TIME	INFOT.	STEPS	MSG	FCE	NZDP
Mini 1 vs. 1	7	UI1	3	3	1202	5	0	8	0	19	0
		UI3	4	6		6	3	8	0	21	0
Mini 2 vs. 2	7	UI1	1	1	522	8	0	8	0	21	0
		UI3	6	9		8	3	7	13	25	0
Mini 3 vs. 3	7	UI1	1	1	372	9	0	8	0	25	1
		UI3	6	10		8	3	8	26	34	0

Tabulka E.6: Souhrnné výsledky 6. experimentu.