Testovanie multiagentových systémov

MAREK MARDIAK

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
xmardiakm[zavináč]stuba[.]sk

Abstrakt. Distribuovaná umelá inteligencia, (Distributed artificial intelligence, DAI), ktorá existuje už dlhšiu dobu ako subdoména oblasti umelej inteligencie, sa zaoberá systémami pozostávajúcimi z mnohých nezávislých entít, ktoré navzájom komunikujú, ovplyvňujú sa a spoločne riešia zadané problémy. Keďže sa tieto multiagentové systémy, resp. distribuované multiagentové systémy, (Distributed multiagent systems, DMAS) uplatňujú na riešenia stále väčšej množiny problémov, dostáva sa do popredia otázka verifikácie a testovania týchto systémov. V tejto práci načrtneme a opíšeme metódu testovania DMAS, konkrétne metódu testovania nežiaduceho emergentného správania sa. Súčasne sa budeme zaoberať využiteľnosťou a slabými stránkami tejto metódy pri testovaní DMAS systémov.

Úvod

V dnešnej dobe prudkého vývoja v oblasti informatiky, techniky, či sa jedná o teoretické poznatky a teórie ešte len čakajúce o uvedenie do reálnej implementácie, alebo empiricky získané poznatky a skúsenosti, oblasť informatiky a informačných technológií čoraz viac zasahuje do každej oblasti nášho života. Keby sme chceli byť konkrétnejší, môžeme tvrdiť, že to, čo nazývame umelou inteligenciou (Artificial Intelligence-AI) je k nám čoraz bližšie. Informačné systémy a celkovo softvér častokrát dnes už zďaleka nie sú statické algoritmy vykonávajúce funkcie implementované počas návrhu, ale dynamické systémy schopné sa v istej miere učiť a hľadať riešenia, dolovať znalosti z obrovského množstva informácií, nachádzať riešenia pre ich doménu problémov, ktoré neboli explicitne implementované. Techniky AI sa stávajú čoraz robustnejšie a komplexnejšie. Multiagentové systémy (MAS) sú subdoménou AI, ktorá si berie za cieľ skúmať ako princípy tvorby komplexných systémov, obsahujúcich mnoho na sebe nezávislých, avšak navzájom komunikujúcich entít, tzv. agentov, tak mechanizmy na koordináciu ich individuálnych správaní sa. Použitie týchto systémov na riešenie mnohých oblastí problémov rastie, avšak zaostáva metodika a metódy ich testovania. Hoci mnohé princípy môžu byť generalizované z metód testovania objektovo-orientovaných systémov, sú tu aspekty, ktoré sú pre túto

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-6.

oblasť rozdielne a vyžadujú znalosť systémov MAS a špecifický návrh postupov a techník ich testovania. Pozrime sa teda bližšie na problematiku testovania MAS, rôzne prístupy a použiteľnosť týchto techník na zaistenie kvality a overenie správnej funkčnosti MAS systémov, menovite projektu RoboCup Soccer Simulator.

Agenty a multiagentové systémy

Keďže sa v tejto práci venujeme testovaniu MAS systémov zo všeobecného hľadiska, zadefinujeme si abstraktne pojmy *Agent* a *Multiagentový systém*, bez implementačných detailov [3].

Agent *Ag* je štvorica *Ag=(Sit, Act, Dat, fg)*. *Sit* je množina situácií, v ktorej sa agent môže nachádzať, *Act* množina akcií, ktorú je schopný vykonať a *Dat* množina možných hodnôt interných dátových štuktúr agenta *Ag*. Agent si nasledujúcu akciu určuje pomocou funkcie *fg: Sit x Dat -> Act*, aplikovanej na súčasnú situáciu a stav interných dátových štruktúr.

Multiagentový systém *MAS* je dvojica *MAS=(A,Env)*, pozostávajúca z množiny agentov *A* a prostredia (environment-*Env)*, v ktorom navzájom komunikujú, a ktoré samé o sebe má vplyv na ich schopnosti a akcie. V systéme RoboCup Soccer Simulator sú naši agenti softvérové entity ovládajúce hráčov, prostredím je simulované 3D hracie ihrisko s prepočítavanou fyzikou a fyzikálnymi parametrami limitujúcimi schopnosti samotných hráčov (ich hmotnosť, energia, pozícia ťažiska atď.).

Prostredie *Env* môžu meniť akcie agentov, môže sa meniť nezávisle, *Env* pozostáva teda z množiny stavov popisujúce zmeny. Až doposiaľ nenachádzame nič nedeterministické na činnosti agentov, avšak situácia v množine *Sit* môže obsahovať pohľad, (projekciu istých atribútov), agenta na aktuálny stav prostredia *Env*, informácie o ostaných agentoch na základe vnemov agenta ako aj špecifické informácie o agentovi, neuložené v dátovej množine *Dat*. Množstvo jednotlivých konkrétnych inštancií týchto premenných, je také veľké, že sa v praxi často používajú klasifikačné algoritmy, ktoré premietnu tieto hodnoty na konkrétne prvky množiny *Sit*. Všimnime si, že nedeterminizmus (vyplývajúci z praktického obmedzenia simulácie všetkých možných inštancií premenných s ktorými agent počas svojej činnosti pracuje, a ktoré sú často ovplyvňované zásahmi „zvonka“ (používateľ, iné softvérové systémy)), nám značne komplikuje testovanie takýchto systémov, hlavne detekciu neexplicitne hľadaných chýb a správaní sa testovaného systému, tzv. emergentných správaní. Spôsob ich vhodného testovania je témou ďalšieho odseku.

Testovanie nevhodného emergentného správania

Dosiahnutie emergentného správania sa skupiny agentov v systéme MAS je ostro sledovaná oblasť výskumu v doméne AI. Výskum sa najviac orientuje na také emergentné správanie, ktoré vytvára znásobujúce sa efekty, ktorých výsledkom je, že skupina agentov dosiahne spoločne cieľ, ktorý je nad úrovňou jednoduchej sumy ich individuálnych schopností. Avšak emergentné správanie sa skupiny agentov často

môže viesť k vyslovene nevhodnej činnosti vyvíjaného systému. Klasické metódy testovania pomocou generovania náhodných sekvencií interakcií tu zlyhávajú, keďže emergentné správanie je často dôsledkom adaptabilných schopností jednotlivých agentov.

V práci [2] je prezentovaný postup detekcie nevhodného emergentného správania. Pozrime sa na jeho hlavné aspekty a možnosti využitia pri testovaní systémov rozsahom obdobných systému RoboCup Soccer Simulator. Hlavnou myšlienkou tohto prístupu je mať skupinu tzv. útočných agentov, konajúcich ako užívatelia a ostatne systémy, s ktorými naši testovaní agenti pri bežnej práci komunikujú. Na tejto skupine útočných agentov sa cez evolučné algoritmy snažíme vyprodukovať také interakčné sekvencie, ktoré sa výsledkom čoraz viac približujú nevhodnému správaniu sa testovaného systému.

Použitie učiacich sa algoritmov na detekciu nevhodného emergentného správania

Majme multiagentový systém *MAStested=(Atested, Env)*, v ktorom chceme zistiť nevhodné emergentné správanie. Použijeme skupinu tzv. útočných agentov *Aattack*, ktorá hrá rolu entít, s ktorými agenti v *Atested* komunikujú v prostredí *Env*. Tieto entity sú často tímoví kolegovia z *Atested*, používatelia testovaného systému alebo iné systémy komunikujúce v *MAStested*. V *Env* tiež máme množinu tzv. hosťujúcich, (bystanders), agentov *Abyst*, ktorí nie sú pod kontrolou nášho testovacieho systému, ani subjektom testovania, ale komunikujú s agentami v *Atested*. Schému systému zobrazuje Obr. 1.

Kľúčovým aspektom je, že správanie sa agentov v *Aattack* je výsledkom istého učiaceho sa postupu, ktorého cieľom je dosiahnutie (alebo aspoň priblíženie sa) k nevhodnému správaniu sa testovaných agentov v *Atested*, cez komunikáciu s nimi, komunikáciu s hosťujúcimi agentami v *Abyst* a interakciu s prostredím *Env*. Zo schémy na Obr. 1. je vidieť že výstupy z tohto učiaceho procesu sú iteratívne vkladané do agentov v *Aattack* istým centrálnym učiteľom (Learner). Samotný učiaci proces však môže byť v samotných agentoch, čím sa môžu dosahovať variabilnejšie výsledky cez distribúciu poznatkov medzi agentami v *Aattack*.

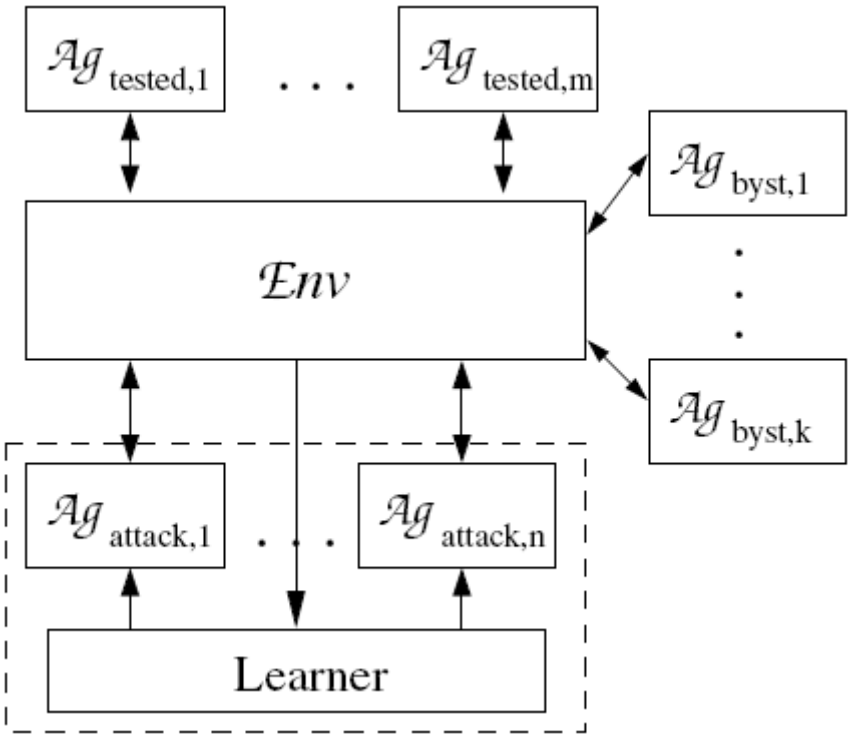
Najdôležitejším a podľa môjho názoru doslova kritickým prvkom tohto prístupu je ohodnotenie, resp. určenie kritérií, pre to, čo je a čo nie je nevhodné emergentné správanie. Systém totiž musí nejako ohodnotiť stavy prostredia *Env*, ktoré sú výsledkom interakcie útočných agentov s testovanými agentami. V práci [2] označujú túto podmienku (funkciu) *Gmerge*. Dôležitosť tohto aspektu pri použiteľnosti tejto metódy v reálnych projektoch postavených na architektúre MAS, rozoberieme v nasledujúcej časti.

Implementácia samotného učiaceho postupu závisí na architektúre samotných testovaných agentov, ich možných inštancií, variability stavov prostredia. Veľmi vhodnou technikou je použitie evolučných algoritmov, pretože vedia účinne nájsť optimálne hodnoty veľmi rozmanitých problémov, bez potreby detailnejšieho popisu situácie, v ktorej sa samotní útoční agenti nachádzajú. Spojenie znalostných algoritmov

a náhodných prvkov, typických pre evolučné algoritmy, vytvára v samotnom procese učenia sa a výberu efekty podobné ľudskej intuícii, čo je tri testovaní veľmi vhodný jav.

Evolučné algoritmy vo všeobecnosti pracujú na množine tzv. indivíduí, prvkov, ktoré sami o sebe reprezentujú adeptov na riešenia daného problému. Jednotlivé riešenia sú vhodne zakódované do jedincov analogicky DNA kódu v prírode. Vhodnou voľbou tzv. fitnes funkcie, ktorá vyberá jedincov do procesu párenia, (čím posúva ich riešenie- DNA do ďalšej generácie a tým bližšie k záverečnému výsledku), ako aj procesmi simulujúcimi mutácie v prírode, sa snažíme nájsť riešenie, (najschopnejších adeptov). Fitnes funkcia reprezentuje vlastne čo o možnom riešení vieme doteraz, a jedincov vyberá teda na základe ich reprezentácie, resp. priblížení sa ku splneniu cieľovej podmienky, stavu ktorý chceme dosiahnuť, v našom prípade splneniu podmienky *Gmerge*. Po mnohonásobne veľa iteráciách voľby jedincov do párenia, vzniku novej generácie, ďalším párením postupne systém konverguje do populácie reprezentujúcej najlepšie nájdené riešenie.

Samozrejme ako som spomenul, kľúčom k úspechu je správna definícia funkcie, ktorú optimalizujeme resp. určuje náš problém, v tomto prípade funkcie *Gmerge.*



Obr. 1. Schéma testovania cez učiacich sa útočných agentov

Úskalia testovania emergentného správania pomocou učiacich sa techník

Ako som spomenul, problém testovania MAS, nielen jeho bežnej funkcionality, ale aj emergentného správania a jeho vhodnosti/nevhodnosti, leží hlavne v značnom nedeterminizme procesov, bežiacich v týchto systémoch. Nemalú úlohu tu zohráva aj fakt, že MAS systémy sú vo svojej podstate systémy reálneho času, kedy je replikovateľnosť scenárov testovania (dôležitá vlastnosť pri detekcii chýb a testovaní softvéru) problematická. Princípom, silou vyššie spomenutého prístupu detekcie nevhodného emergentného správania je tiež nedeterminizmus samotného učiaceho procesu. S trochou nadhľadu to môžeme prirovnať k likvidácii veľkých lesných požiarov vytvorením požiarov pracujúcich proti nim, využitiu rovnakej črty (pri požiaroch vlastnosti spaľovania, resp. potreby materiálu na proces horenia, v našom prípade nedeterminizmu spoločnému ako testovanému tak testovaciemu systému).

V prípade zabezpečenia a kontroly kvality softvéru sa však pracuje s obmedzenými zdrojmi (ľudia, čas na testovanie) a očakávajú sa konkrétne výsledky. Tak ako vedia MAS prísť s nevhodným správaním, čo nebol určite zámer tvorcov takého systému, tak môžu rovnakým nedostatkom trpieť aj analogicky postavené systémy (obsahujúce útočných agentov, klasifikáciu, evolučné algoritmy a iné prístupy z domény AI) detekujúce tieto nežiaduce vlastnosti.

Veľmi veľa záleží na vhodnej voľbe limitnej podmienky pre detekciu nevhodného správania, *Gmerge*. Ak však vieme popísať kritéria nevhodného správania sa veľmi presne, je na mieste otázka, prečo vôbec používať formu detekcie na báze nedeterminizmu. Ak sú naše znalosti o kritériách nevhodného správania sa u MAS systému slabšie, testovací systém môže nesprávne vyhodnotiť niektoré činnosti nášho systému.

Pozrime sa na použitie, klady ale aj úskalia nedeterminizmu pri evolučnom prístupe. V systéme RoboCup Soccer Simulator, simulovanej lige robotického futbalu, sa v jeho jednej forme, RoboCup3D, snažil tím Neurotics (detailná dokumentácia ich MAS systému je v [1]) riešiť problém vstávania robota (agenta) v simulovanom 3D prostredí, berúceho do ohľadu zákony bežnej fyziky. Pomocou evolučných algoritmov a vhodnou formou reprezentácie výsledného riešenia, kde jedinec je definovaný genetickou informáciou, ktorú tvorí chronologická postupnosť základných pohybov robota (pohybov jednotlivých kĺbov indexovaných časom) a vhodnej definície fitnes funkcie (rozdiel maximálnej dosiahnutej výšky hlavy a výšky hlavy vo vzpriamenej polohe, čas za ktorý sa do tejto výšky dostal, ako dlho v nej zotrval, množstvo energie potrebnej na postavenie sa) hľadali optimálny proces vstávania robota. Výsledkom bola stratégia, kedy robot vyskočil do výšky a stabilizoval sa pri následnom páde na zem, kde sa snažil (úspešne) ostať vo vzpriamenej polohe. Robot však pri takomto postupe veľmi často explodoval, čím simulačný systém nahradzoval prehrievanie sa v reálnom svete. Evolučný prístup však napriek tomu vyhodnotil toto správanie ako najviac optimálne, i keď pre potreby systému (futbalový tím) je strata hráča kritickou chybou. Na voľbe vhodnej fitnes (*Gmerge*) funkcie teda veľmi záleží, a jej nájdenie a špecifikácia môže byť veľmi náročnou činnosťou.

Podľa môjho názoru, má však prístup testovania emergentného správania sa pomocou učiacich sa techník viac pozitív ako negatív a pri nájdení vhodného kompromisu medzi silou (mierou presnosti špecifikácie) limitnej podmienky *Gmerge* a použitia klasických prístupov testovania softvéru (testovanie aplikačných jednotiek, Unit testing) môžeme dosiahnuť uspokojivé výsledky vzhľadom na kvalitu a pokrytie výsledného testovania MAS systémov, ako aj na spotrebu zdrojov (ľudia, čas) v tejto fáze tvorby a manažmentu softvérového projektu.

Záver

V práci sme sa zamerali na testovanie multiagentových systémov MAS, konkrétne detekciu nežiadaného emergentného správania sa týchto systémov. Venovali sme sa metóde detekcie tohto javu s využitím evolučného prístupu, detailnejšie popísanej v [2]. Rozobrali sme použiteľnosť danej metódy pri MAS systémoch rozsahu RoboCup Soccer Simulator a analyzovali jej silné a slabé stránky ako aj aspekty použiteľnosti v rámci manažmentu a kontroly kvality softvérových systémov.

Použitá literatúra

1. Braniša, G., Čimo, P., Katona, A ., Kolesár, T., Labuda, J., Šimon, J.: Robocup-tretí rozmer(Dokumentácia k projektu). Fakulta informatiky a informačných technológií STU, Bratislava, 2007.
2. Denzinger, J., Kidney, J.:Evaluating Different Genetic Operators in the Testing for Unwanted Emergent Behavior using Evolutionary Learning of Behavior. University of Calgary, Calgary, 2006.
3. Stone, P., Veloso, M.: Multiagent Systems: A Survey from a Machine Learning Perspective. Kluwer Academic Publishers, 2000.

Annotation

Testing of mutiagent systems

Distributed artificial intelligence, (DAI), which is a subdomain of artificial intelligence, (AI), works with independent systems that communicate and influence each other and by doing so solve defined problems. Because of the breadth of problems these systems are being used to solve, a question of verification and testing of these systems arise. In this thesis we will explain the DMAS testing method, specifically the testing of unwelcomed emergent behavior. Also we will discuss usability issues and the weak points of this method used in testing DMAS systems.