Seleção de Casos de Teste para Agentes Racionais

Francisca Raquel de Vasconcelos Silveira[1]

Gustavo Augusto Lima de Campos[1]

Mariela I. Cortés[1]

Resumo: Apesar dos referenciais disponíveis para orientar o projeto de agentes racionais, existem poucas técnicas de testes propostas para validá-los. Sabe-se que essa validação depende dos casos de teste selecionados, os quais devem gerar informações que permitam a identificação dos componentes na estrutura do programa do agente artificial testado que estão causando o desempenho insatisfatório. Este trabalho propõe uma abordagem que visa contribuir com o processo de teste destes programas. A abordagem envolve uma formulação matemática para o problema de seleção de casos de teste e um agente orientado por utilidade para resolver o problema. Os primeiros experimentos realizados visaram avaliar a abordagem selecionando casos de teste para programas de agentes reativos simples em um ambiente parcialmente observáveis.

Palavras-chaves: Agentes racionais, seleção de casos de teste, teste de programas artificiais.

Abstract: Despite the theoretical references available to guide the design of rational agents, there are few test techniques proposed to validate them. It is known that this validation depends on the selected test cases, which should arrange for information concerning the components in the structure of the agent tested program that are underperforming. This paper proposes an approach that aims to contribute to the process of testing these programs. The approach involves a mathematical formulation to the problem of selecting cases and an agent oriented by utility to solve the problem. The first experiments aimed to evaluate the approach by selecting test cases for simple reactive agent programs in partially observable environments.

Keywords: Rational agents, selection of test cases, test of artificial agents.

1 Universidade Estadual do Ceará, UECE, Caixa Postal 60.740-903
{raquel_silveira@ifce.edu.br}, {gustavo, mariela @larces.uece.br}

1 Introdução

Um agente é uma entidade capaz de perceber seu ambiente de tarefa por meio de sensores e agir neste por meio de atuadores. O comportamento de um agente pode ser descrito em termos matemáticos por uma função do agente capaz de mapear qualquer sequência de percepções específica para uma ação. Esta função do agente é implementada concretamente pelo programa do agente, que é executado em uma arquitetura adequada (dispositivo de computação com sensores e atuadores físicos). Idealmente, os agentes racionais, aqui decompostos em um programa de agente e em uma arquitetura, devem agir visando alcançar o melhor resultado esperado, avaliado de acordo com uma medida de desempenho [10].

As aplicações de sistemas baseados em agentes racionais abrangem as mais diversas áreas como, por exemplo, referentes à assistência ao paciente, simulação de campo de batalha, detecção de intrusão, ferramentas de recomendação, gerenciamento de processos de negócios, jogos e batalhas de robôs, controlador de naves espaciais e aeronaves, etc. [5]. Estas aplicações requisitam uma confiabilidade maior do que aquelas que funcionam, simplesmente, reagindo ao controle imediato de seus usuários. Apesar de existirem esforços no sentido de facilitar o desenvolvimento de sistemas baseados em agentes racionais, ainda existem poucos métodos e técnicas para testar a eficiência destes sistemas [2].

Existem quatro tipos de teste aplicáveis aos sistemas de em agentes racionais [4]. Primeiro, o teste de cada unidade de agente visa avaliar suas funcionalidades internas e capacidades para realizar objetivos, perceber e agir no ambiente. Segundo, o teste de integração verifica se um agente que foi integrado previamente em um grupo trabalha corretamente com outro agente que ainda será integrado. Terceiro, o teste de sistema visa garantir que todos os agentes no sistema trabalham em conjunto conforme esperado. Finalmente, o teste de aceitação busca garantir que o sistema multiagente funcionará corretamente no ambiente de execução e verificar se o mesmo realiza os objetivos do projeto.

Por outro lado, sendo um produto da engenharia, nos testes caixa branca faz-se um exame interno do detalhe procedimental do agente. Nos testes caixa preta examinam-se os aspectos funcionais do agente através de uma interface, sem se preocupar com sua estrutura interna. Os principais níveis de teste dos produtos de engenharia são: caixa branca/unitário para explorar a menor unidade do projeto; caixa branca/integração para testar a integração dos módulos; caixa preta/sistema para avaliar o software em busca de falhas; e caixa preta/aceitação para simular as operações de rotina do sistema com os usuários finais [7] e [9].

Devido às propriedades peculiares dos agentes racionais (propriedades reativas, de memória, orientação por metas e por utilidade, e de aprendizagem) e de seus ambientes de tarefa, o teste unitário de cada agente em um sistema é uma tarefa não trivial. A maioria dos trabalhos consiste em adaptações das técnicas de testes de software convencional. No caso dos agentes racionais, sabe-se que estas adaptações devem buscar avaliar a racionalidade das ações e dos planos executados pelo agente em seu ambiente de tarefa [10].

Algumas abordagens focam a produção de artefatos de teste para dar suporte às metodologias de desenvolvimento de sistemas de agentes [4]. O pressuposto na maioria dos

trabalhos é que uma boa avaliação do agente depende dos casos de testes selecionados. Bons casos de teste devem providenciar a geração de informações sobre o desempenho insatisfatório dos componentes na estrutura do agente e do funcionamento destes componentes de maneira integrada [12]. Neste contexto algumas abordagens focam a seleção automática de casos para testar os agentes. Vale ressaltar as abordagens evolucionárias para gerar casos com níveis de dificuldade crescentes [5].

Este artigo apresenta uma abordagem que visa contribuir com o processo de desenvolvimento de testes caixa preta/unitário dos programas dos agentes artificiais componentes dos sistemas. Os pressupostos são que qualquer tipo de teste depende dos casos de teste selecionados, os quais devem gerar informações que permitam identificar os componentes na estrutura do programa do agente artificial testado estão causando desempenho insatisfatório; e, como nem sempre os melhores casos estão disponíveis a priori e, dependendo do ambiente de tarefa do agente, pode existir uma grande quantidade de casos a serem observados, a seleção de um conjunto de casos de teste é um problema de busca em um espaço de estados composto por uma grande família de conjuntos de casos possíveis.

Assim, a abordagem proposta consiste em uma formulação matemática para o problema de seleção de casos de teste de agentes e em um agente orientado por utilidade, que emprega aspectos presentes em metaheurísticas baseadas em população para encontrar conjuntos de casos de teste satisfatórios, isto é, descrições de ambientes específicos em que as histórias associadas do programa de agente testado em seu ambiente obtiveram baixo desempenho. Durante o processo de busca de um conjunto de casos de teste, o agente utiliza um protocolo de interação entre o programa do agente testado e outro programa ambiente de tarefa para realizar simulações, avaliar e anotar as histórias correspondentes. Os primeiros experimentos realizados visaram avaliar a abordagem selecionando casos de teste para programas de agentes reativos simples em um ambiente parcialmente observável.

O artigo é organizado em mais quatro seções principais. A Seção 2 discorre sobre os principais conceitos relacionados aos agentes racionais e aos testes destes agentes, e sobre alguns trabalhos relacionados. A Seção 3 a visão geral da abordagem e, mais especificamente, uma formulação matemática para o problema de seleção de casos de teste e o esboço do agente orientado por utilidade para resolver o problema. A Seção 4 apresenta os resultados dos primeiros experimentos produzidos utilizando a abordagem para o teste de agentes reativos. Finalmente, a Seção 5 apresenta as conclusões e os próximos trabalhos a serem desenvolvidos para a evolução da abordagem proposta.

2 O Teste de Agentes Racionais

O agente racional seleciona suas ações objetivando o melhor resultado possível, ou na presença de incertezas, o melhor resultado esperado conforme uma medida de desempenho estabelecida para avaliar seu comportamento. Conceber agentes racionais para atuar em ambientes de tarefa complexos é uma tarefa não trivial. O trabalho da Inteligência Artificial consiste em projetar o programa do agente, que implementa a função do agente e será

executado em alguma arquitetura. Dependendo do ambiente, o projeto do agente pode ser idealizado considerando quatro tipos de programas agentes: reativos simples, reativos baseados em modelos, baseados em objetivos e baseados em utilidade. Em ambientes onde o agente não conhece os estados possíveis, nem os efeitos das suas ações, a concepção de um agente racional pode requisitar um programa agente com capacidade de aprendizagem [10].

Os quatro tipos de programas agentes podem ser subdivididos em três subsistemas de processamento de informação principais. O primeiro, o subsistema de percepção, mapeia uma informação sobre percepção (P) em uma representação abstrata (Estado) útil para o agente, **ver**: P→Estado. O segundo, o subsistema de atualização de estado interno, mapeia a representação da percepção atual e a informação sobre o estado interno (EI) mantido pelo agente em um novo estado interno, **próximo**: Estado x EI → EI. Por último, o subsistema de tomada de decisão, mapeia uma informação sobre estado interno em uma ação possível (A), **ação**: EI → A [11]. Por exemplo, para o caso do programa agente reativo simples, a função ação seleciona ações baseando-se na percepção atual, mapeada pela função ver, e em um conjunto de regras no formato condição-ação. Pesquisas têm sido realizadas sobre os diferentes aspectos relacionados ao teste de agentes, entretanto ainda é requerido um processo de este estruturado para agentes ainda é requerido.

A abordagem de teste para agentes apresentada em [2] é especificado um processo estruturado para a geração de testes que complementa a metodologia Tropos e reforça a relação mútua entre a análise de objetivos e testes [3]. O processo fornece uma forma de derivar casos de teste a partir dos objetivos dos agentes. Essa abordagem não contempla: a noção de agentes racionais, uma medida de avaliação de desempenho e uma simulação que possa monitorar o comportamento do agente ao executar as ações para realizar seus objetivos. Em [4] é proposto um método de análise orientada a objetivos, visando um processo de testes sistemático e abrangente para agentes que engloba o processo de desenvolvimento do agente de acordo com a metodologia Tropos a partir dos requisitos iniciais até a implantação [3]. É proposta uma metodologia de produção de artefatos de testes a partir das especificações dos agentes e do *design*, e usa estes artefatos para detectar problemas. Os casos de teste são gerados automaticamente e evoluem guiados por mutação e função de qualidade.

Uma abordagem evolucionária para os testes de agentes é proposta em [5] e aplica o recrutamento dos melhores casos de teste para evoluir os agentes. Para cada agente é dado um período experimental para execução de diferentes níveis de dificuldade de testes. Os agentes são recrutados apenas quando passam pelo critério de qualidade. Tanto em [4] quanto em [5] o agente testado é implementado de acordo com o modelo BDI (*Belief, Desire* e *Intention*). Apesar de considerar a noção de agentes racionais, a abordagem não emprega esquemas de simulação para monitorar o comportamento do agente ao executar as ações.

3 Abordagem Proposta

A abordagem proposta está fundamentada na noção de agentes racionais, na utilização de casos de teste gerados de acordo com os objetivos na medida de avaliação de desempenho

do agente testado, na simulação da avaliação de desempenho das interações do agente testado com seu ambiente e em estratégias de busca local multiobjetivo orientada por utilidade para encontrar casos de teste e histórias em que o agente não foi bem avaliado.

3.1 Visão Geral da Abordagem

A abordagem (Figura 1) considera que, além do projetista do agente testado, existem quatro programas de agentes envolvidos: o programa do agente artificial a ser testado, Agent, concebido pelo projetista; um programa de agente ambiente de tarefa, Amb; um programa de agente para a seleção de casos de testes, Thestes; e um agente monitorador, ProMon, que recebe dados de Thestes e disponibiliza para o projetista informações sobre o desempenho de Agent e sobre os componentes na sua estrutura interna que estão com falhas.

O projetista é responsável por conceber o programa do agente artificial Agent e a medida de avaliação de desempenho, e alimentar outras informações necessárias para o agente Thestes iniciar o processo de teste de Agent em Amb. O agente Thestes consiste em um agente de resolução de problemas de seleção de casos de teste que realiza busca local no espaço de estados de casos de teste orientado por utilidade. Este agente envia para o agente ProMon um conjunto de soluções eficientes determinada pela estratégia de busca multiobjetivo, ou seja, descrevendo casos de teste em que Agent possui o comportamento mais inadequado, um conjunto de histórias correspondentes e seus valores de utilidade.

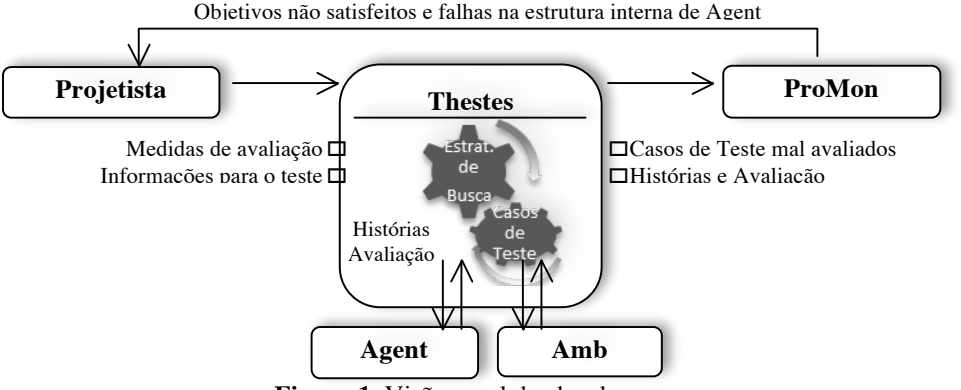


Figura 1. Visão geral da abordagem

O artigo coloca em destaque apenas o agente para a seleção de conjuntos de casos de teste. Assim, nesta apresentação, as informações na Figura 1 que são enviadas para o agente ProMon, devem ser analisadas pelo projetista. A versão atual do agente ProMon incorpora funcionalidades semelhantes às que o projetista utiliza para gerar conclusões. Este agente conhece o tipo de programa de agente sobre o qual Agent foi concebido e, ao receber as informações enviadas por Thestes, identifica para o projetista: os objetivos na medida de avaliação que não estão sendo satisfeitos adequadamente por Agent, os episódios nas histórias de Agent em Amb que são falhas, os episódios ideais correspondentes, e os módulos de processamento da informação de Agent (ver, próximo, ação) que estão gerando as falhas.

3.2 Problema de Seleção de Casos de Teste de Agentes

Um agente é racional se for capaz de tomar decisões corretas, ou seja, que realizem seus objetivos em um ambiente de tarefa, ou, quando não for possível realizar todos os objetivos, tomar as decisões de maior sucesso, definido de acordo com alguma medida de avaliação de desempenho. Esta medida considera um ou mais aspectos do ambiente. A Tabela 1 especifica uma medida de avaliação de desempenho para o caso de um programa aspirador de pó (uma versão adaptada de [10]) que deve limpar um ambiente e maximizar os níveis de limpeza do ambiente e de energia em sua bateria ao final da tarefa.

Tabela 1. Medida de avaliação desempenho

$\mathbf{P^K}$	$\mathbf{A^K}$	$\mathbf{av_E(P^K, A^K)}$	$\mathbf{av_L(P^K, A^K)}$
..., L, ...	Asp	-1.0	0.0
..., L, ...	Dir, Esq, Ac, Ab	-2.0	1.0
..., L, ...	N-op	0.0	0.0
..., S, ...	Asp	-1.0	2.0
..., S, ...	Dir, Esq, Ac, Ab	-2.0	-1.0
..., S, ...	N-op	0.0	-1.0

A primeira coluna descreve parte da percepção do agente (sala limpa = L ou sala suja = S) em cada episódio possível. A segunda coluna descreve as ações possíveis nesses episódios (aspirar = Asp, direita = Dir, esquerda = Esq, acima = Ac, abaixo = Ab e não operar = N-op). As duas últimas colunas são associadas aos objetivos energia e limpeza, respectivamente, duas funções escalares (av_E e av_L) para medir o desempenho do agente em cada episódio. O projetista do agente deve considerar este tipo de medida e conceber o programa visando maximizar seu desempenho nas histórias do agente no ambiente. Do ponto de vista da abordagem isto equivale ao seguinte problema de seleção de casos de teste (PMO).

A formulação do problema de seleção de casos de testes considera que, se o programa agente for inadequado ao seu ambiente de tarefa então as funções objetivos de inadequação, ou seja, as funções escalares na medida de avaliação modificadas pelo sinal de menos (–), serão maximizadas. Entretanto, pode não existir um conjunto que seja ótimo em todas as funções e, neste caso, a tarefa consiste em encontrar um conjunto de casos satisfatório, ou seja, em que o desempenho do programa de agente no ambiente é insatisfatório e que permita ao projetista perceber as propriedades limitativas do programa.

Seja:

- **Agent**: um programa agente racional a ser testado;
- **Amb**: programa ambiente capaz de interagir com Agent por meio de um protocolo;
- **ProtocolInteração**: uma descrição do protocolo de interação entre Agent e Amb;
- **Ω**: um conjunto de descrições de ambientes de tarefa factíveis de instanciar em Amb e testar Agent;
- **P(Ω)**: subconjuntos de ambientes possíveis de serem descritos em Ω;
- **CasosTEST ∈ P(Ω)**: um subconjunto de descrições de ambientes casos de teste específico no conjunto P(Ω), onde:
  - **$Caso_i$ ∈ CasosTEST**: uma descrição específica de ambiente no subconjunto CasosTEST;
- **H(CasosTEST) ∈ P($(PxA)^{NInt}$)**: conjunto de histórias de comprimento NInt de Agent em Amb considerando ProtocolInteração e todos os casos em CasosTEST tal que ∀i □ {1, ..., NCasos}, ∀t □ {1, ..., NInt}:
  - **h($Caso_i$) ∈ $(PxA)^{NInt}$**: história de comprimento NInt de Agent em Amb correspondente ao $Caso_i$ ∈ CasosTEST;
  - **Ep^t(h($Caso_i$)) ∈ PxA**: episódio na interação t, t ≤ NInt, da história de Agent em Amb correspondente ao caso $Caso_i$ ∈ CasosTEST;
- **f_{ad}(H(CasosTEST)) = (f_1(H(CasosTEST)), ..., f_M(H(CasosTEST))) ∈ R^M**: um vetor de M funções objetivo (implícitas) na medida de avaliação de desempenho estabelecida pelo projetista (M ≥1), mede a adequação de Agent a Amb considerando um conjunto de histórias H(CasosTEST), onde, ∀m □ {1, ..., M}:

$$f_m\left(H(CasosTEST)\right) = \frac{1}{NCasos} \sum_{i=1}^{NCasos} Av_m\left(h(Caso_i)\right)$$

são as funções que o projetista busca maximizar, mede o nível de adequação de Agent a Amb, no que diz respeito à realização do m-ésimo objetivo na medida de avaliação de desempenho, considerando todas as histórias em H(CasosTEST); e ∀i □ {1, ..., NCasos}:

$$Av_m\left(h(Caso_i)\right) = \sum_{p=1}^{NInt} av_m\left(Ep^p\left(h(Caso_i)\right)\right)$$

tal que $av_m(Ep^p(h(Caso_i)))$ é o valor de recompensa/penalização no objetivo m atribuído pela medida de avaliação ao episódio p da história associada ao $Caso_i$ ∈ CasosTEST;

- **f_{inad}(H(CasosTEST)) = (– f_1(H(CasosTEST)), ..., – f_M(H(CasosTEST))) ∈ R^M**: um vetor de M objetivos associado ao vetor f_{ad}(H(CasosTEST)), mede a inadequação de Agent a Amb considerando as histórias em H(CasosTEST).

Problema:

'maximizar' f_{inad}(H(CasosTEST))
s.a: CasosTEST □ P(Ω) e H(CasosTEST) □ P($(PxA)^{NInt}$)

3.3 O Agente Thestes

A noção de agentes é uma ferramenta disponível bastante útil para analisar e conceber sistemas e foi empregada na concepção do sistema de resolução do problema de seleção de casos testes. Do ponto de vista de projetos de agentes únicos, a literatura sobre o assunto disponibiliza diversas arquiteturas abstratas de agentes e diversos tipos de programas de agentes que podem ser concretizados para resolver problemas em ambientes de tarefas que requisitam uma tomada de decisão não trivial. O agente Thestes foi concebido como um agente de resolução de problemas de seleção de caso de testes. Seu esqueleto fundamenta-se na estrutura do programa de agentes orientado por utilidade especificada por [10] e na arquitetura abstrata do agente com estado interno [11].

A Figura 2 ilustra a estrutura do agente de resolução de problemas Thestes. Seu subsistema de percepção, ver, mapeia as informações necessárias ao teste de Agent em uma representação computacional, Estado^K, adequada ao processamento dos outros subsistemas: (Agent, Amb, ParâmetrosBUSCA, ParâmetrosSimulação). O subsistema de atualização de estado interno, próximo, armazena as informações em Estado^K, considera os parâmetros em ParâmetrosSimulação e gera um conjunto inicial CasosTEST de maneira aleatória. Considerando o estado interno atualizado, a função ação de Thestes inicia um processo de busca local baseada em populações e orientada por uma função utilidade visando encontrar uma ação satisfatória Ação^K para ser enviada ao projetista ou seja, descrições de ambientes específicos em que as histórias associadas de Agent em Amb possuem baixo desempenho.

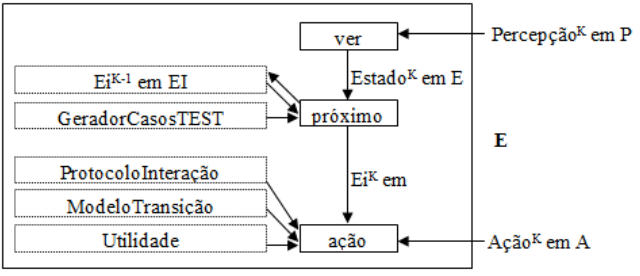


Figura 2. Estrutura do programa agente Thestes

A função ação utiliza informações a respeito de um modelo de transição de estados para gerar novos casos de teste a partir de CasosTEST, e o protocolo de interação e a função utilidade para, respectivamente, obter as histórias correspondentes aos casos de teste e avaliar o desempenho de Agent nestas histórias. O modelo de transição modifica os conjuntos de casos de teste, considerando os casos de teste em uma solução corrente, Pop^t = CasosTEST, os valores de desempenho correspondentes, medidos por uma função Utilidade para gerar novos casos de teste, Pop^{t+1}. O modelo de transição genérico considera os NCasos em um conjunto Pop^t corrente e escolhe: (a) os casos de teste que serão modificados, e (b) as mudanças que serão realizadas nestes casos.

Thestes conhece Agent e Amb, bem como o protocolo ProtocoloInteração. A função ação considera estas informações e ParâmetrosSimulação no processo de tomada de decisão. Assim, foi concebido um mecanismo de interação que simula as interações entre Agent e Amb, quando Amb é inicializado com informações de um caso de teste ($Caso_i \in CasosTEST$) e anota os episódios ($Ep^p(h(Caso_i)) \in PxA$) da história ($h(Caso_i) \in (PxA)^{NInt}$). Thestes intercepta as mensagens de Agent e Amb, a fim de obter as histórias dos casos de teste. A Figura 3 ilustra o funcionamento deste mecanismo.

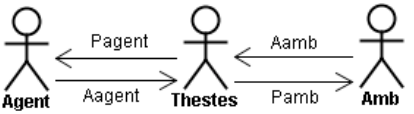


Figura 3. Simulação das interações Agent-Amb

O mecanismo alimenta Amb com informações em P_{Amb} a respeito de um caso de teste específico pertencente ao conjunto solução corrente CasosTEST. O programa Amb armazena essas informações e envia em A_{Amb} as informações de estado corrente para Agent. Antes de repassar estas informações para Agent, o mecanismo anota estas informações. Ao perceber estas informações em P_{Agent}, Agent processa-as e seleciona uma ação que é enviada em A_{Agent} para o mecanismo. Ao receber estas informações sobre ação, o mecanismo encerra a anotação de um episódio da história, envia em P_{Amb} as informações sobre ação para Amb e inicia outro ciclo de interação, que deve ser repetido até a anotação de uma história completa.

Quanto à função utilidade em Thestes, no problema de seleção multiobjetivo formulado, a função inadequação a ser maximizada foi definida como $f_{inad}(H(CasosTEST)) = (-f_1(H(CasosTEST)), ..., -f_M(H(CasosTEST))) = Y \in R^M$. Assim, vale notar que f_{inad}: $P(\Omega) \square Y \subset R^M$ e que Y é parcialmente ordenado de acordo com a relação de ordem '$\preceq$', ou seja: em geral, $\neg\exists CasosTEST^* \in P(\Omega)$ e $Hists(CasosTEST^*) \in P((PxA)^{NInt}) \ni \forall$ $CasosTEST \in P(\Omega)$ e $Hists(CasosTEST) \in P((PxA)^{NInt})$, $f_{inad}(Hists(CasosTEST^*)) \geq f_{inad}(Hists(CasosTEST))$. Assim, para decidir entre dois vetores comparáveis $y^1, y^2 \in Y$, a estrutura de preferências de Thestes considera a relação característica clássica (**S**), composta da união das relações binárias de indiferença **I** e de preferência estrita **P** tal que apenas uma das sentenças é verdadeira para Thestes: (1) y^1 **I** y^2, ou seja, Thestes é indiferente entre y^1 e y^2; (2) y^1 **P** y^2, ou seja, Thestes prefere y^1 a y^2; e (3) y^2 **P** y^1, ou seja, Thestes prefere y^2 a y^1. Assim, o problema de seleção formulado foi caracterizado em Thestes como: determinar $y^* \in Y \ni \forall y \in Y, y^*$ **P** y.

Para resolver o problema, os julgamentos de Thestes foram representados numericamente por meio de uma função utilidade Utilidade: $Y \square R$, ou seja, que associa um real Utilidade(y) a cada $y \in Y \subset R^M$ tal que: (1) y^1 **I** $y^2 \leftrightarrow U(y^1) = U(y^2)$; (2) y^1 **P** $y^2 \leftrightarrow U(y^1) > U(y^2)$; (3) y^1 **S** $y^2 \leftrightarrow U(y^1) \geq U(y^2)$. Supondo-se que a função Utilidade é preferencialmente independente, ou seja, o grau de utilidade de um objetivo independe dos valores assumidos pelos demais, esta primeira concretização incorporou em Thestes uma função Utilidade no formato linear:

$$\text{Utilidade}\left(f_{inad}(H(\text{CasosTEST}))\right) = -\sum_{m=1}^{M} w_m * f_m(H(\text{CasosTEST}))$$

tal que $w_m \geq 0$, m = 1, ..., M. Assim, o problema de seleção de casos de teste foi reformulado:

'maximizar' Utilidade(f_{inad}(H(CasosTEST)))
s.a: CasosTEST □ P(Ω) e H(CasosTEST) □ P((PxA)^{NInt})

Ou seja, considerando que Y = f_{inad}(P(Ω)) é a representação do mapeamento de P(Ω) no espaço dos objetivos: Y = {y ϵ R^M| y = f_{inad}(H(CasosTEST))), CasosTEST ϵ P(Ω) e H (CasosTEST) ϵ P((PxA)$^+$)}; o problema pode ser estabelecido como:

'maximizar' Utilidade(y)
s.a: y ∈ Y

Essa reformulação permite que Thestes estruture suas preferências através de curvas de indiferença ou isopreferência onde, para um determinado nível de utilidade, uma curva de indiferença apresenta os valores de inadequação associados aos objetivos em que o agente é indiferente, representam combinações de valores de inadequação nos objetivos que auferem o mesmo nível de utilidade. Assim, o agente não tem preferência entre uma combinação e a outra, já que cada uma provê um mesmo nível de utilidade. De posse das informações, Thestes envia ao projetista quatro informações importantes em Ação^K: (1) o conjunto CasosTEST solução para o problema de seleção com os melhores casos encontrados, (2) as histórias correspondentes de Agent em Amb, (3) os valores de desempenho em cada um dos objetivos, e (4) outras informações relevantes para o diagnóstico de problemas em Agent.

4 Avaliando o Agente Thestes

Nesta seção é ilustrado o funcionamento do esqueleto de Thestes, resolvendo um problema de seleção para testar um agente aspirador de pó em dois programas agentes: reativo simples e reativo com estado interno, ambos com percepção local e avaliados em um ambiente com várias salas considerando os atributos energia e limpeza (Tabela 1).

4.1 O Ambiente de Tarefa e o Agente em Teste

A tarefa de Thestes consiste em selecionar um conjunto de ambientes que sejam satisfatórios para testar um programa agente aspirador de pó com regras condição-ação. Um ambiente difere de outro quanto à localização e à quantidade de salas sujas. Todo ambiente é parcialmente observável, ou seja, pressupõe-se que o aspirador percebe o ambiente, mas sua função ver consegue mapear apenas o estado da sala que o agente está.

O programa agente aspirador de pó reativo simples (Asp_RS_Parcial) foca na seleção das ações com base na percepção atual, ignorando o histórico de percepções obtido, em um ambiente parcialmente observável, ou seja, a função ver permite perceber apenas o estado, sujo ou limpo, da sala em que o agente está. Assim, este primeiro programa foi concebido de

maneira muito simples, mas útil para uma primeira ilustração e avaliação da abordagem. A Figura 4 apresenta as regras condição-ação desse agente.

se estado da sala é S **então faça** Asp
se estado da sala é L **então faça** movimento aleatório (Ac, Esq, Dir, Ab)

Figura 4. Regras condição-ação do agente aspirador reativo simples

O segundo programa agente testado foi concebido de acordo com a estrutura do agente reativo com estado interno (Asp_REi_Parcial) considerando que o ambiente é parcialmente observável. Este agente possui um estado interno que armazena o histórico das percepções obtidas e uma ação é selecionada levando em considerando esta informação. A Figura 5 apresenta as regras condição-ação desse agente.

se estado da sala é S **então faça** a ação Asp
se estado da sala é L e NaoVisitou(norte) **então faça** a ação Ac
se estado da sala é L e NaoVisitou(sul) **então faça** a ação Ab
se estado da sala é L e NaoVisitou(leste) **então faça** a ação Dir
se estado da sala é L e NaoVisitou(oeste) **então faça** a ação Esq
se estado da sala é L e visitou todas **então faça** uma ação aleatória

Figura 5. Regras condição-ação do agente aspirador com estado interno

4.2 Concretização de Thestes para o Problema

O esqueleto de Thestes pressupõe a existência de um conjunto CasosTEST corrente contendo NCasos de teste, que é uma solução inicial gerada pela função próximo. A função ação do agente considera quatro componentes principais: (1) um ModeloTransição de estados que opera sobre CasosTEST para gerar novos conjuntos, (2) uma função utilidade para avaliar os conjuntos gerados, (3) uma estratégia para selecionar entre os conjuntos avaliados um novo conjunto corrente mais útil, e (4) um teste para o novo conjunto CasosTEST. Para os componentes (1), (3) e (4) o estudo empregou noções dos algoritmos genéticos [1]. Para o componente (2), empregou uma função Utilidade no formato linear, conforme descrito na Seção 3.

No contexto do GA, CasosTEST, contendo as descrições de ambientes de tarefa compostos de *n x n* salas, foi representado por uma população, onde cada indivíduo codifica um ambiente e cada gene codifica o estado da sala, em termos de sujeira. O ModeloTransição baseado em GA considera CasosTEST como uma população que está apta a passar pelas etapas de seleção de pares (empregado o método da roleta), cruzamento e mutação, que provocam a evolução e que permitem à função ação realizar simulações, avaliar a utilidade dos indivíduos e compor uma nova população melhor. Visando prevenir a perda do melhor caso de teste encontrado empregou-se elitismo nas modificações do conjunto CasosTEST. O melhor caso de teste é aquele onde o agente apresenta o pior desempenho na avaliação.

4.3 Experimento com o Agente Thestes

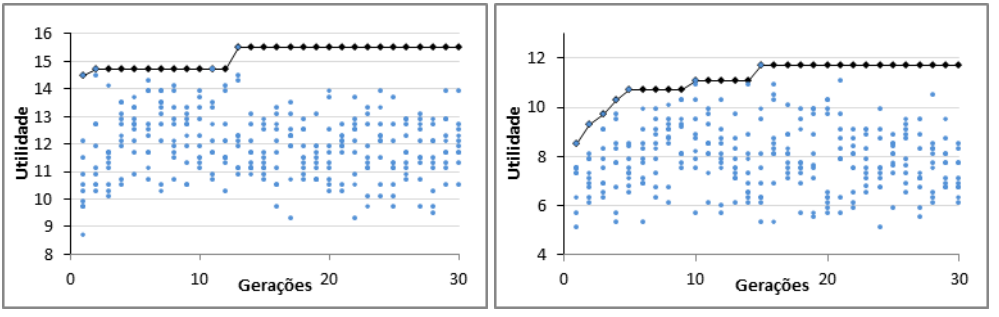
Esta seção apresenta um experimento realizado com Thestes. ParâmetrosBUSCA descrevem o operador e a probabilidade de cruzamento (O_{Cros} e P_{Cros}) entre os casos de teste, o operador e a probabilidade de mutação (O_{Mut} e P_{Mut}), o número máximo de execuções ($K_{máx}$), e o valor de utilidade máxima que pode ser alcançada por uma história ($U_{máx}$). As informações sobre $K_{máx}$ e $U_{máx}$ definem a condição de parada no mecanismo de teste. As informações em ParâmetrosSimulação descrevem a quantidade (NCasos) e o tamanho (n^2) de ambientes em CasosTEST, o número máximo de interações entre Agent e Amb em qualquer simulação (N_{int}), ou seja, o número máximo de episódios em cada história, e o número de simulações realizadas em um mesmo ambiente (Ns). A Tabela 2 apresenta estas informações.

Tabela 2. Informações em ParâmetrosBUSCA e ParâmetrosSimulação

ParâmetrosBusca						**ParâmetrosSimulação**			
O_{Cros}	P_{Cros}	O_{Mut}	T_{Mut}	$K_{máx}$	$U_{máx}$	NCasos	n^2	N_{int}	Ns
SinglePointCrossover	0,9	*Uniform*	0,6	30	100000	10	25	25	5

O número máximo de interações (N_{int}) de Agent com Amb, a serem simuladas em cada um dos 10 casos (NCasos) na população, é 25. O valor de utilidade máxima ($U_{máx}$) é alto, e a condição de parada no mecanismo de teste da estratégia de busca é definida considerando 30 ciclos de execuções da função ação ($K_{máx}$). Para cada caso de teste na população foram realizadas cinco simulações (Ns). A Figura 6 apresenta a função Utilidade no formato linear com pesos iguais para limpeza e energia, respectivamente, na medida de avaliação, ou seja, $w_L = w_E = 0,5$ para o agente Asp_RS_Parcial em (a) e Asp_REi_Parcial em (b). Os pontos não lineares representam os casos de teste em CasosTEST em cada geração. Os pontos lineares identificam o melhor caso de teste na geração.

No caso do agente reativo simples com observabilidade parcial (Figura 6 (a)), o melhor caso de teste foi obtido na décima quarta geração, com valor de utilidade igual a 15,5. Este caso serviu como referencial para os casos em CasosTEST nas gerações posteriores com valores de utilidade menores que o melhor. No caso do agente reativo com estado interno (Figura 6 (b)), o melhor caso de teste foi obtido na décima quinta geração, com valor de utilidade igual a 11,7.

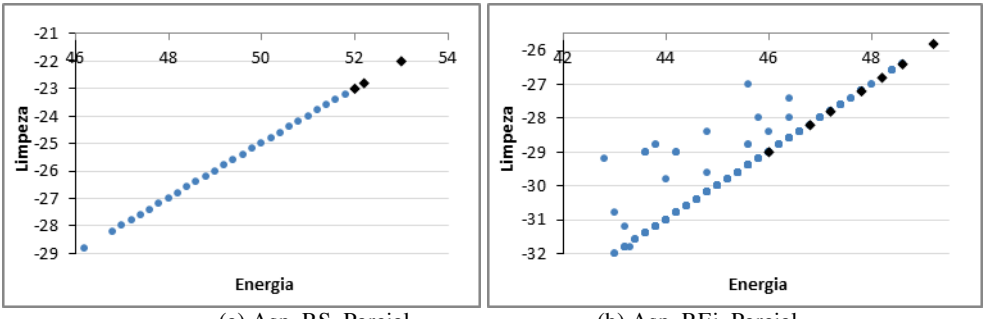


(a) Asp_RS_Parcial (b) Asp_REi_Parcial

Figura 6. Valores de utilidade de 10 casos CasosTEST em 30 gerações

Considerando os valores de utilidade dos melhores casos de teste, Thestes detectou que o agente mais inadequado para o ambiente parcialmente observável é o reativo simples, quando comparado com o agente reativo com estado interno. Neste caso, a anotação das salas que foram visitadas pelo subsistema de atualização de estado interno (próximo) do agente baseado em modelos possibilitou a concepção de um subsistema de tomada de decisão (ação) mais refinado para o agente reativo com estado interno. O conjunto de regras componentes deste subsistema evitou o retorno desnecessário às salas que já visitadas em interações anteriores com o ambiente, o que não foi possível para o agente reativo simples.

A Figura 7 apresenta os valores das funções de inadequação de limpeza e energia associados aos 10 casos em CasosTEST nas 30 gerações. Comparando os resultados entre os agentes, considerando o caso de teste em CasosTEST com maior valor de utilidade conforme Figura 6, é possível observar que para ambos os atributos, energia e limpeza, apresentados na Figura 7, o agente reativo simples com observabilidade parcial possui um comportamento mais inadequado, com $-f_E = 53{,}0$ e $-f_L = -22{,}0$, enquanto o agente reativo com estado interno possui comportamento menos inadequado, com $-f_E = 49{,}2$ e $-f_L = -25{,}8$. Isto pode ser justificado devido às propriedades internas de cada um dos agentes.



(a) Asp_RS_Parcial (b) Asp_REi_Parcial

Figura 7. Valores de Inadequação de Limpeza e Energia dos 300 casos

A princípio, como os dois objetivos na medida de avaliação de desempenho tem o mesmo valor de importância ($w_L = w_E = 0{,}5$) na função utilidade, a abordagem privilegia os casos em que Agent tem um comportamento mais inadequado em termos do consumo de energia que em termos de limpeza, quase que maximizando este nível de inadequação conforme indicado em todas as gerações. Isto se justifica, pois a medida de avaliação de desempenho pontua negativamente em termos de energia todos os episódios possíveis para Agent, devido ao agente sempre gastar energia ao executar suas ações, e positivamente em termos de limpeza os episódios. A Tabela 3 ilustra cinco episódios de uma simulação da interação de Agent em Amb, no ambiente que obteve melhor valor médio de utilidade para o agente reativo simples.

Tabela 3. História parcial de Agent em Amb

K	P^K	A^K	$-av_E(P^K, A^K)$	$-av_L(P^K, A^K)$
1	..., L, ...	Ab	2.0	-1.0
2	..., L, ...	Dir	2.0	-1.0
3	..., L, ...	Ab	2.0	-1.0
4	..., S, ...	Asp	1.0	-2.0
5	..., L, ...	Esq	2.0	-1.0

O ambiente de tarefa selecionado foi constituído de 25 salas com a seguinte configuração: [[L, L, L, L, S], [L, L, S, S, L], [L, S, S, S, S], [L, L, L, S, S], [L, L, S, S, S]]. O valor de utilidade foi U = 15,5 e os valores de inadequação: $-f_E$ = 49,0 e $-f_L$ = -26,0. Os demais episódios da história seguem o mesmo padrão. Conforme esperado, o agente aspirador é mais adequado ao ambiente considerando o critério de limpeza que o critério de energia. Uma análise breve das regras condição-ação do aspirador de pó confirmará esta proposição. A história obtida por Asp_REi_Parcial segue o mesmo padrão.

Assim, conforme esperado, o agente aspirador de pó reativo simples com observabilidade parcial apresenta o pior desempenho na avaliação, visto que ao realizar uma análise breve nas regras condição-ação do agente se confirmará que, por limitação da arquitetura, as mesmas não consideram nas condições em seus antecedentes aspectos relacionados aos critérios de energia e de limpeza. Visto que o aspirador foi concebido como um programa reativo simples, pouco pode ser feito para melhorar seu desempenho, sendo necessário realizar uma extensão em sua estrutura visando ampliar a observabilidade do ambiente, ou acomodar um estado interno, o que permitirá ao agente economizar energia ao perceber em seu histórico de percepções que uma determinada sala já foi visitada.

5 Considerações Finais

Considerando que o agente racional deve ser capaz de realizar seus objetivos, testes adequados devem ser desenvolvidos para avaliar as ações e os planos executados pelo agente na realização destes objetivos. Para a realização dos testes em agentes racionais é necessário que técnicas que tratem da natureza peculiar do agente sejam aplicadas. A abordagem apresentada considera que no caso dos agentes racionais, em que a medida de avaliação de desempenho é estabelecida pelo projetista, envolve vários objetivos que podem ser conflitantes. Os resultados obtidos demonstram que a abordagem proposta é eficaz, pois consegue gerar informações relevantes que permitem ao projetista raciocinar sobre o desempenho insatisfatório dos componentes na estrutura interna do programa do agente em teste. Mas, ainda é necessário que se realize, um estudo de caso com os agentes baseados em objetivos e em utilidade.

Referências

[1] HOLLAND, J. Adaptation in natural and artificial systems. *University of Michigan Press*, 1975.

[2] HOUHAMDI, Z. Test Suite Generation Process for Agent Testing. In: *Indian Journal of Computer Science and Engineering (IJCSE)*, v. 2, n. 2, 2011.

[3] MYLOPOULOS J.; CASTRO J. Tropos: A Framework for Requirements-Driven Software Development. *Information Systems Engineering: State of the Art and Research Themes*, Lecture Notes in Computer Science, Springer, 2000.

[4] NGUYEN, C. D. Testing Techniques for Software Agents. *PhD Dissertation*. University of Trento 2008.

[5] NGUYEN, C. D.; PERINI, A.; TONELLA, P.; MILES, S.; HARMAN, M.; LUCK, M. Evoluctionary Testing of Autonomous Software Agents. *Autonomous Agents and Multi-Agent Systems*. v. 25, n. 2, p. 260-283, 2012.

[6] NGUYEN, C.; PERINI, A.; BERNON, C.; PAVÓN, J.; THANGARAJAH, J. Testing in multi-agent systems. *Springer*. v. 6038, p. 180-190, 2011.

[7] PRESSMAN, R.S. Engenharia de Software. 6.ed. Rio de Janeiro: McGraw-Hill, 2006.

[8] POUTAKIDIS, D.; WINIKOFF, M.; PADGHAM, L.; ZHANG, Z. Debugging and Testing of Multi-Agent Systems using Design Artefacts. *Springer Science Business Media*, LLC, 2009.

[9] ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. Qualidade de software – Teoria e prática. São Paulo: Prentice Hall, 2001.

[10] RUSSELL, S.; NORVIG, P. *Inteligência Artificial*: uma abordagem moderna. 2 ed. São Paulo: Prentice-Hall, 2004.

[11] WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 200 2.

[12] ZINA, H. *Test Suite Generation Process for Agent Testing*. Indian Journal of Computer Science and Engineering (IJCSE), v. 2, n. 2, 2011.