Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Bruna Thalenberg
Eduardo Freire Carvalho de Lima

Jogos de Segurança e Proteção Ambiental

São Paulo
Dezembro 2021

Jogos de Segurança e Proteção Ambiental

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: José Coelho de Pina

São Paulo
Dezembro de 2021

"[Technology] makes progress possible. But technology does not free us of the need for leadership; it makes leadership all the more important. So let us not forget that technology by itself cannot absolve us of our political responsibility to ensure that we use it wisely and efficiently for the good of society everywhere."

(Kofi Annan, Keynote Speech at the Massachusetts Institute of Technology, 2015)

Resumo

Bruna Thalenberg e Eduardo Freire Carvalho de Lima. **Jogos de Segurança e Proteção Ambiental**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

Os jogos de segurança de Stackelberg são uma aplicação da Teoria dos Jogos voltada para a otimização da alocação de recursos de segurança. Após o sucesso de algumas implementações iniciais, como no patrulhamento de segurança do aeroporto de Los Angeles, em 2007, grupos de pesquisadores passaram a estudar a aplicação desses jogos em situações de proteção da natureza, como no combate à pesca ilegal e ao desmatamento. Neste trabalho, apresentamos os fundamentos teóricos que nos levam dos Jogos de Stackelberg aos Jogos de Segurança Ambiental, passando pelos Jogos de Segurança. Apresentamos abordagens algorítmicas para a sua solução e detalhamos a implementação proposta por Yang et al. [22] para o monitoramento da caça ilegal nas florestas de Uganda.
Palavras-chave: Jogos Sequenciais, Jogos de Segurança, Jogos Bayesianos.

Abstract

Bruna Thalenberg e Eduardo Freire Carvalho de Lima. **Security Games and Environmental Protection**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo São Paulo, 2021.

Stackelberg Security Games are an application of Game Theory to resource optimization, specifically for optimizing the allocation of limited security resources. After the success of initial implementations in the Los Angeles Airport in 2007, researchers began to study the application of this framework towards nature protection. In this report, we present the fundamentals of game theory from Stackelberg Games to Green Security Games, we review algorithmic approaches to solve these games and we detail the implementation proposed by Yang et al. [22] to prevent ilegal poaching in a national park in Uganda.
Keywords: Sequential Games, Security Games, Bayesian Games.

Sumário

Introdução

Os jogos de segurança de Stackelberg são uma aplicação da Teoria dos Jogos [10] voltada para a otimização da alocação de recursos de segurança. Nesses jogos, há dois jogadores: um defensor e um atacante. O defensor possui um número limitado de patrulhas e busca defender um conjunto também limitado de alvos. Um alvo está protegido se está sendo monitorado por pelo menos uma patrulha. O atacante, por sua vez, deseja selecionar um alvo para atacar que não esteja protegido. Alguns exemplos de aplicações reais são o patrulhamento de aeroportos contra atentados, a proteção de fronteiras e o monitoramento de portos [20].

Os jogos de Stackelberg são diferentes de jogos típicos por serem não-simultâneos, uma vez que um dos jogadores, o defensor, compromete-se inicialmente com uma estratégia de alocação de suas patrulhas e, em seguida, o outro jogador, atacante, decide seu movimento, que pode ser o alvo que será atacado, ou não atacar nenhum alvo. A estratégia do defensor pode envolver estratégias mistas ou aleatorizadas para a decisão de posicionamento das patrulhas.

Após o sucesso de algumas implementações iniciais, como no patrulhamento de segurança do aeroporto de Los Angeles, em 2007 [14], grupos de pesquisadores passaram a estudar a aplicação desses jogos em situações de proteção da natureza, como no combate à pesca ilegal e ao desmatamento [4]. Esse subconjunto de jogos é conhecido como Jogos de Segurança Verdes (*Green Security Games*) [4], e há, entre eles, algumas particularidades em sua modelagem por conta das características dos ataques – frequentes e repetidos, com baixo custo no caso de perda, diferentemente de ataques terroristas, em geral únicos e mais longamente planejados [20].

A preservação e uso sustentável dos ecossistemas terrestres não apenas é fundamental para a sobrevivência de animais e plantas, mas também é crucial para a manutenção da vida humana na Terra, em face da crise climática e ambiental que enfrentamos. De acordo com a IPBES, Plataforma Intergovernamental Científico-Política sobre a Biodiversidade e Serviços Ecossistêmicos, a natureza é essencial para a existência humana e para a qualidade de vida, pois tem um papel essencial em fornecer alimentação, energia, medicamentos, recursos genéticos e uma variedade de materiais fundamentais para o bem estar físico das pessoas [19]. Por exemplo, mais de dois bilhões de pessoas dependem de biocombustíveis como fonte primária de energia, e estima-se que quatro bilhões de pessoas dependam primariamente de medicamentos naturais ou sintéticos inspirados pela natureza para a manutenção de sua saúde [19]. Através dos processos ecológicos e evolutivos, a natureza sustenta a qualidade do ar, a água fresca e os solos dos quais a humanidade depende. Os ecossistemas marinhos e terrestres são os únicos sumidouros de carbono [*carbon sinks*], sequestrando 5.6 gigatoneladas de carbono por ano, equivalentes a aproximadamente 60% das emissões antropogênicas anuais [19].

No entanto, a natureza ao redor do globo tem sido alterada significativamente por diversas ações humanas, com a maioria dos indicadores dos ecossistemas e biodiversidade mostrando rápido declínio: 75% da superfície terrestre foi significativamente alterada, 66% da área do oceano está sofrendo impactos crescentes e cumulativos, e 85% da área pantanosa

desapareceu [19]. Nos trópicos, 32 milhões de hectares de floresta nativa ou em recuperação foram perdidos entre 2010 e 2015 [19]. Aproximadamente metade da cobertura viva de coral em recifes foi perdida desde 1870, com as perdas sendo aceleradas nas décadas recentes por conta da mudança climática [19]. A média de espécies nativas na maioria dos principais biomas terrestres caiu em ao menos 20% desde 1900, e aproximadamente 25% das espécies e plantas conhecidas estão ameaças, sendo que em torno de um milhão de espécies estão a beira da extinção e provavelmente serão extintas nas próximas décadas [19]. Essa perda de biodiversidade implica em um sério risco para a segurança alimentar global ao diminuir a resiliência de diversos sistemas agrícolas contra ameaças como pestes, patógenos e a mudança climática [19].

Sua importância é tal que

> "proteger, recuperar e promover o uso sustentável dos ecossistemas terrestres, gerir de forma sustentável as florestas, combater a desertificação, deter e reverter a degradação da terra e deter a perda de biodiversidade"

é um dos Objetivos de Desenvolvimento Sustentável (ODS) da Agenda 2030 da ONU [21][1].

De acordo com o Painel Intergovernamental sobre Mudanças Climáticas (IPCC), uma ampla gama de adaptações e respostas de mitigação, como a preservação e restauração de ecossistemas naturais, a conservação da biodiversidade, a redução da competição por terras, a gestão de incêndios e a gestão dos solos, tem o potencial de contribuir positivamente para o desenvolvimento sustentável e para o aumento das funções e serviços ecossistêmicos. [1]

Neste trabalho, apresentamos os fundamentos teóricos que nos levam dos Jogos de Stackelberg aos Jogos de Segurança Ambiental, passando pelos Jogos de Segurança. Apresentamos abordagens algorítmicas para a sua solução, e analisamos uma implementação de uma dessas soluções para o monitoramento da caça ilegal nas florestas de Uganda com base nos resultados apresentados por Yang et al [22]. Esperamos, com isso, contribuir para o desenvolvimento dos estudos de jogos de segurança no Brasil, além de fornecer uma ferramenta prática que possa auxiliar as agências ambientais na preservação de nossas matas, mares e rios. O texto assume que o leitor possua conhecimento dos conceitos compreendidos em um curso introdutório de teoria dos jogos. No entanto, caso esclarecimentos ou detalhamentos sejam necessários, recomendamos a consulta ao livro-texto de Morris [10].

O texto está dividido em 5 capítulos. No capítulo 1, introduzimos os conceitos de Teoria dos Jogos que são necessários para o entendimento do restante do texto. São abordados os conceitos de Jogos, Estratégias, Pagamentos, Equilíbrios e o Teorema Minimax. No capítulo 2, passamos ao domínio de Jogos Sequenciais, e apresentamos os Jogos de Stackelberg, o conceito de Equilíbrio de Stackelberg, e como o Teorema Minimax relaciona-se com ele. Na sequência, no capítulo 3, abordamos os Jogos de Stackelberg Bayesianos, passando pela definição de Jogos de Informação Incompleta, Jogos Bayesianos e a Transformação de Harsanyi, para finalmente alcançarmos os Jogos de Stackelberg Bayesianos. No capítulo 4, apresentamos alguns dos algoritmos utilizados para a solução de jogos de segurança, como o DOBSS e o ERASER-C, e suas aplicações para o domínio da segurança de infraestrutura, na proteção de aeroportos e aviões. Enfim, no capítulo 5, adentramos o subdomínio da segurança ambiental e abordamos suas particularidades, além de detalhar a aplicação do caso PAWS para a proteção do Parque Nacional Rainha Elizabeth, em Uganda.

[1] Todos os Objetivos, bem como todas as metas do objetivo citado acima, podem ser conferidos nos Apêndices A e B

Capítulo 1

Jogos Simultâneos

Neste capítulo, apresentaremos definições básicas de Teoria dos Jogos que serão importantes para o entendimento do texto.

1.1 Jogos

Formalmente, um **jogo** se refere a qualquer situação de conflito que envolva dois ou mais agentes inteligentes, denominados **jogadores**, tomando decisões racionais. Um jogador é considerado **inteligente** se tem conhecimento das regras do jogo e é capaz de tomar decisões baseadas em seu conhecimento. É costumeiro dividir os jogos entre **jogos cooperativos** e **jogos não cooperativos**, sendo que os primeiros admitem algum arranjo entre os jogadores na escolha de sua estratégia. Neste texto, iremos nos concentrar em jogos não cooperativos.

Para modelarmos um jogo, precisamos dizer quem são os seus participantes, como cada um destes pode agir, quais são os possíveis resultados destas ações e qual a ordem de preferências entre estes resultados para cada jogador.

Para os participantes, ou jogadores, serão utilizados os rótulos $1, 2, \ldots, n$. Dependendo do problema sendo modelado, os jogadores poderão ser chamados de **atacantes** ou de **defensores** (ou **defesa**).

As possíveis ações de cada jogador serão descritas através de um conjunto, e cada elemento desse conjunto receberá o nome de **estratégia**. Finalmente, cada jogador terá suas preferências de resultados especificadas por uma função que, dependendo da aplicação, será chamada de **custo**, **pagamento** ou **utilidade**. Cada jogador toma uma decisão ou age em um determinado momento. Esse momento é o que chamamos de **turno** ou vez do jogador. Turnos podem ser simultâneos, em que todos os jogadores realizem suas ações concomitantemente, ou não-simultâneos, em que há uma ordem predeterminada para que cada jogador realize sua ação. Chamamos de **rodada** um conjunto de turnos. Por exemplo, no jogo de par-ou-ímpar, os turnos são simultâneos, e uma rodada é constituída quando os dois jogadores apontam seus dedos indicando dígitos um a cinco. Já no jogo Banco Imobiliário, os turnos são não-simultâneos, e cada jogador deve lançar os dados, avançar o número de casas correspondentes e realizar ou receber pagamentos, a depender da casa resultante. Uma rodada, neste jogo, é constituída quando todos os jogadores completam seu turno, e a "vez" de jogar volta para o primeiro jogador.

Para a modelagem de um jogo, é costumeiro utilizar a **forma normal de jogo**:

$$\Gamma_n = (N, \{S_i\}_{i \in N}, \{U_i\}_{i \in N})$$

em que:

- $N = \{1, \ldots, n\}$ é o conjunto de jogadores,
- Para i, $i = 1, 2, \ldots, n$, S_i é o conjunto de estratégias do jogador,
- Para i, $i = 1, 2, \ldots, n$, $U_i : S_1 \times \cdots \times S_n \to \mathbb{R}$ é a função de pagamento do jogador i.

Além da forma normal, também é possível utilizar a **forma extensiva**, em que o jogo é representado por uma árvore. Neste contexto, uma **árvore** é um grafo orientado em que, com excessão feita a um vértice especial r chamado de **raiz** da árvore, todo vértice é ponta final de um arco. A raiz não é ponto final de arco algum. Se uv é um arco da árvore, então dizemos que u é o pai de v e que v é filho de u. No exemplo abaixo, temos que r é pai de a e a é filho de r, bem como a é pai de c e c é filho de a. Um vértice sem filhos é chamado de vértice terminal, ou **folha**, e para uma determinada árvore T, o conjunto de seus vértices terminais será denotado TERM(T). Note que, em uma árvore, existe um único caminho da raíz até cada vértice.

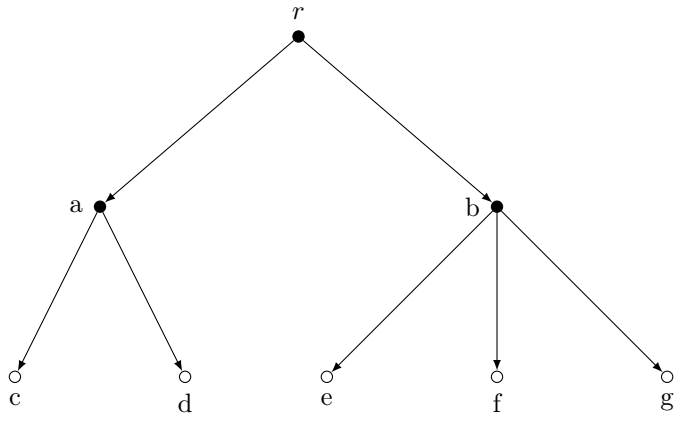


Figura 1.1: *Árvore*

Uma **árvore de jogo** é constituída por quatro elementos que denotaremos (N, T, S, U). O primeiro elemento é um conjunto N finito com n elementos que, assim como na forma normal, representa o conjunto de jogadores. O segundo elemento é uma árvore $T = (V, A)$, em que V é seu conjunto de vértices e A é seu conjunto de arcos. O terceiro elemento é uma função $S : V \setminus \text{TERM}(T) \to N$, cujo domínio é o conjunto dos vértices não terminais de T. Isso significa que cada vértice pertence a algum jogador do conjunto N. Por fim, U representa a função de pagamento da forma $U : \text{TERM}(T) \to \mathbb{R}^n$. Ela irá determinar o ganho (ou a perda) de cada jogador ao final do jogo.

Uma árvore de jogo é um modelo do seguinte jogo: o jogador de N que possui a raiz começa o jogo escolhendo um filho deste vértice. Em seguida, o jogador que possui este vértice escolhido realiza a escolha de um dos seus filhos e assim sucessivamente, até que se atinja um vértice terminal, e o ganho de cada jogador será apurado de acordo com a função U.

Uma sequência de escolhas até um determinado vértice de T é um histórico do jogo até ali. Como, para cada vértice, só existe um único caminho da raiz a ele, então identificaremos cada vértice como um histórico do jogo.

1.2 Estratégias

A estratégia de um jogador pode ser definida como um plano de ações para tornar o jogo benéfico a ele. As estratégias podem ser divididas em puras e mistas.

Definição 1.1 *Uma **estratégia pura** do jogador i é sua definição antecipada de qual será sua ação nos momentos do jogo em que lhe caiba realizar uma jogada. O conjunto de todas as estratégias puras possíveis para o jogador i é denotado por $S_i = \{s_i^1, s_i^2, \ldots, s_i^{m_i}\}$.*

Dada uma árvore de jogo (N, T, S, U), e considerando que V_i é o conjunto de vértices que pertencem ao jogador i, uma **função de escolha** para o jogador i é uma função $E : V_i \to V$ que satisfaça a condição $(v, E(v)) \in A$, ou seja, E seleciona, para cada vértice v pertencente a i, um filho desse vértice de acordo com a árvore T. Uma n-tupla $(E_1, \ldots E_n)$ de funções de escolha é chamada de **perfil de jogo**, que contém, para cada jogador i, sua função de escolha E_i.

Um perfil de jogo determina um caminho da raiz até um ponto terminal da árvore: sabendo que a raiz r pertence a algum jogador k, a função de escolha E_k escolhe um vértice $v = E_k(r)$, que ou é terminal, ou pertence a algum jogador l. Neste caso, a função de escolha E_l escolhe um outro vértice $u = E_l(v)$ e assim sucessivamente até chegarmos a um vértice terminal t em que os ganhos serão apurados pela função de pagamento, $U(t)$. Dessa forma, cada perfil $(E_1, \ldots, E_n)$ determina um único caminho da raiz até um ponto terminal e, portanto, um vetor de pagamentos de $\mathbb{R}^n$ que também será chamado $U(E_1, \ldots, E_n)$.

Uma vez que a função de escolha de um jogador determina sua estratégia, também podemos denotar o mesmo perfil de jogo por

$$s = (s_1, s_2, \ldots, s_n), s_i \in S_i; (i = 1, 2, \ldots, n),$$

em que s_1 em S_1 é a estratégia escolhida pelo jogador 1 e assim por diante.

A notação s_{-i} é usada para representar o perfil de jogo s sem a estratégia s_i, ou seja, $(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. Assim, se s_i' está em S_i, então (s_i', s_{-i}) é uma abreviatura de $(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$. Isso permite que escrevamos $s = (s_i, s_{-i})$ como abreviatura para o perfil do jogo $(s_1, s_2, \ldots, s_n)$.

Utilizando essa notação, em um jogo de estratégias puras, temos que $U_i(s_i, s_{-i})$ é a função de pagamento calculada para o jogador i em função do perfil de jogo jogado pelos n jogadores participantes.

1.3 Equilíbrio de Nash

O objetivo de cada jogador é maximizar seu pagamento esperado durante o jogo, definindo sua melhor resposta. Uma estratégia s_i' em S_i é chamada de **melhor resposta** do jogador i para o perfil de estratégias s se i não pode aumentar seu pagamento alterando apenas a sua estratégia,

$$U_i(s_i', s_{-i}) \geq U_i(s_i, s_{-i}) \forall s_i \in S_i$$

No equilíbrio de Nash, todos os jogadores estão jogando sua melhor resposta

Definição 1.2 *Um perfil de jogo $s = (s_1, \ldots, s_n)$ é chamado de **equilíbrio de Nash** ou de ponto de equilíbrio se, para qualquer jogador $i \in N$ temos que $U_i(s_1, \ldots, s_n) \geq U_i(s_1, \ldots, t_i, \ldots, s_n)$ para todo $t_i \in S_i$. Neste caso, denotamos o perfil por $s^* = (s_1^*, \ldots, s_n^*)$.*

Na prática, o equilíbrio de Nash implica que cada jogador está obtendo o melhor resultado possível para si, de forma que, caso mude sua jogada, obterá um pagamento esperado pior.

A **solução de um jogo** será entendida como a previsão de como ele será jogado, considerando as melhores respostas para cada jogador.

1.4 Dilema dos prisioneiros

Um exemplo clássico em teoria dos jogos para ilustrar os conceitos apresentados acima é o **Dilema dos prisioneiros**. A situação tipicamente é descrita da seguinte forma: Dois prisioneiros são mantidos em celas separadas e o promotor do caso oferece a cada um algumas condições conforme descrito a seguir.

- Caso ele testemunhe contra o comparsa e este não testemunhar contra ele, sua pena será de 1 ano de prisão, cabendo a seu colega cumprir 10 anos.
- Caso o comparsa também testemunhe contra ele sua pena será de 5 anos.
- Se, todavia, ambos se recusarem a testemunhar um contra o outro, ambos passarão dois anos na cadeia.

Este é um jogo com dois jogadores: cada um dos prisioneiros. Cada jogador (ou prisioneiro) possui duas estratégias: testemunhar (T) contra o outro ou não (N). O pagamento será o número de anos na prisão – ou seja, o pagamento é negativo. A função de pagamento está expressa na forma da tabela 1.1. Essa forma de expressar os pagamentos possíveis de um jogo é bastante comum, e é chamada de **matriz de pagamento do jogo**, ou matriz de jogo.

	N	T
N	$-2, -2$	$-10, -1$
T	$-1, -10$	$-5, -5$

Tabela 1.1: *Matriz de pagamento do dilema dos prisioneiros*

Note que, caso houvesse comunicação entre os prisioneiros, talvez eles escolhessem a estratégia (N, N). No entanto, como o jogo não é cooperativo, eles não podem ter certeza da escolha dessa estratégia, já que há um prêmio por delação. Note também que, se um escolhe a estratégia N, isso pode lhe resultar em dez anos de cadeia.

O perfil (T, T), em que os dois prisioneiros testemunham um contra o outro, é um perfil de equilíbrio. Com este perfil, o pagamento do prisioneiro 1 é 5 anos de cadeia. Caso ele mude de estratégia e o segundo prisioneiro mantenha a dele o novo perfil seria (N, T) e, neste caso, o primeiro prisioneiro pegaria 10 anos de cadeia, o que é pior. A mesma coisa vale para o segundo prisioneiro.

Como expusemos anteriormente, é claro que o perfil (N, N) pagaria mais a cada jogador do que cada um recebe do perfil de equilíbrio. Mas, neste caso, cada jogador poderia melhorar ainda mais seu pagamento caso ele desviasse da estratégia (N, N) e o outro mantivesse sua jogada. Logo, (N, N) não se configura como um perfil de equilíbrio.

1.5 Estratégias mistas

É muito comum que um jogo não possua um perfil de equilíbrio de estratégias puras. Nesses casos, para determinar as melhores respostas para cada jogador e, portanto, determinar a solução do jogo, podemos imaginar que o jogo irá se repetir diversas vezes, com os jogadores escolhendo estratégias diferentes em cada repetição. Podemos determinar antes com que proporção usaremos cada uma das estratégias puras, ou seja, com que probabilidade a escolheremos. Chamamos essa distribuição de proabilidades sobre as estratégias puras de **estratégias mistas**.

Para ilustrar, suponhamos um jogo de dois jogadores, que, para clareza, chamaremos de Alice e Bob. Os conjuntos de estratégias puras de Alice e Bob serão, respectivamente,

$$S_1 = \{s_1, \ldots, s_n\} \text{ e } S_2 = \{f_1, \ldots, f_m\}.$$

O conjunto das estratégias mistas de Alice será

$$M_1 = p = \{(p_1, \ldots, p_n) \in \mathbb{R}^n : \sum p_i = 1, p_i \geq 0\}$$

e o conjunto das estratégias mistas de Bob será

$$M_2 = q = \{(q_1, \ldots, q_m) \in \mathbb{R}^m : \sum q_j = 1, q_j \geq 0\}$$

A interpretação dessas estratégias mistas é de que p_i é a probabilidade de Alice escolher a estratégia pura s_i, e de maneira similar para Bob, com q_i sendo a probabilidade de que ele escolha a estratégia f_i. Note que uma estratégia pura pode ser representada como uma estratégia mista com probabilidade 1 sobre uma única estratégia e 0 sobre todas as outras estratégias.

Formalmente, definimos:

Definição 1.3 *Seja G um jogo de n jogadores com os conjuntos de estratégias $S_1, \ldots S_n$. Uma **estratégia mista** para o jogador i é um vetor de probabilidades $\vec{p_i} = (p_i(s))_{s \in S_i}$. A entrada $p_i(s)$ é interpretada como a probabilidade de que i jogue a estratégia pura $s \in S_i$*

Tome como exemplo um jogo de pedra, papel e tesoura[1] em que o vencedor é escolhido em uma única rodada. As estratégias puras possíveis seriam, então: jogar pedra, jogar papel, ou jogar tesoura. Já uma estratégia mista poderia ter a forma: jogar pedra com 30% de probabilidade, jogar papel com 20% de probabilidade e jogar tesoura com 50% de probabilidade. Esta estratégia seria $p = (0.3, 0.2, 0.5)$.

Se um ou mais jogadores adotarem estratégias mistas, os pagamentos que ocorrerão de fato uma vez que se realize o jogo dependem de quais estratégias puras foram selecionadas em uma específica instância. Assim, a quantia que de fato é utilizada para o estudo de estratégias mistas é o **pagamento esperado**, que é um cálculo ponderado entre a probabilidade das estratégias e seus pagamentos.

Definição 1.4 *O **pagamento esperado do jogador** i no perfil de jogo (p_i, p_{-i}) em que $p_i(s)$ representa a estratégia mista escolhida pelo jogador i, é calculado por*

$$\pi_i(\vec{p_1}, \ldots, \vec{p_n}) = \sum (p_1(s_1) \times \ldots \times p_n(s_n)) U_i(s_1, \ldots, s_n)$$

no qual a soma é calculada com todas as escolhas de $s_i \in S_i, 1 \leq i \leq n$.

[1] Pedra ganha de tesoura, papel ganha de pedra e tesoura ganha de papel. O jogo é tipicamente jogado em rodadas, de turnos simultâneos, mas também há a forma do duelo único.

1.6 Teorema Minimax

Na área de inteligência artificial, jogos sequenciais recebem o nome de **problemas de busca adversarial**. Para esta seção, nos baseamos no livro de Russell e Norvig [16]. Segundo os autores, a transformação de um jogo em um problema de busca adversarial se dá a partir das seguintes definições:

- S_0: **estado inicial**, que especifica o estado do jogo antes de qualquer jogada
- JOGADOR(s): define qual jogador irá realizar a jogada/movimento no estado s.
- AÇÕES(s): função que retorna o conjunto de movimentos possíveis no estado s.
- RESULTADO(s, a): o **modelo de transição**, que define o resultado de a ser o movimento a partir do estado s.
- TESTE-TERMINAL(s): predicado que é verdadeiro caso o jogo tenha terminado e falso em caso contrário. Estados em que o jogo termina são chamados **estados terminais**.
- PAGAMENTO(s, p): função de pagamento que define o valor numérico final para um jogo que termina no estado s para um jogador p.

O estado inicial, a função AÇÕES e a função RESULTADO definem a árvore de um jogo. Chamamos de **árvore de busca** um corte da árvore de jogo completa que examina vértices suficientes para permitir que um jogador decida qual a melhor jogada a ser feita.

Em um problema de busca tradicional, a solução ótima seria uma sequência de ações que leva ao estado objetivo. Em uma busca adversarial, a solução ótima deve encontrar uma **estratégia de contingência**, que leva em consideração as jogadas do oponente. Considerando as definições de resposta ótima e equilíbrio de Nash apresentadas anteriormente, chamamos de **valor maximin** o maior valor que um jogador pode obter sem ter conhecimento das ações dos outros jogadores. Simetricamente, o **valor minimax** é o menor valor que os outros jogadores podem forçar um jogador a receber, sem ter conhecimento das jogadas dele. Von Neumann, em 1928 [11], demonstrou a equivalência entre os valores minimax e maximin, que ficou conhecida como **Teorema minimax**:

Teorema 1.5 (Teorema Minimax) *Sejam $X \subset \mathbb{R}^n$ e $Y \subset \mathbb{R}^m$ convexos compactos. Se $f : X \times Y \to \mathbb{R}$ é uma função contínua que é côncava-convexa, isto é, $f(\cdot, y) : X \to \mathbb{R}$ é côncava para y fixo e $f(x, \cdot) : Y \to \mathbb{R}$ é convexa para x fixo, então:*

$$\max_{x \in X} \min_{y \in Y} f(x, y) = \min_{y \in Y} \max_{x \in X} f(x, y)$$

Por extensão, em um jogo de dois jogadores, chamaremos um deles de jogador MAX e um deles de jogador MIN. Dada uma árvore de jogo, a estratégia ótima pode ser determinada pelo valor minimax de cada vértice, que denotaremos MINIMAX(N). O valor minimax de um vértice é o pagamento (para MAX) de cada estado, assumindo que os dois jogadores farão jogadas ótimas a partir daquele estado até o término do jogo. O valor minimax de estados terminais é sua utilidade ou pagamento e, frente a uma escolha, MAX move-se para o estado de maior valor, enquanto MIN move-se para o estado de menor valor. Vejamos o exemplo retirado de Russell e Norvig [16] na figura 1.2.

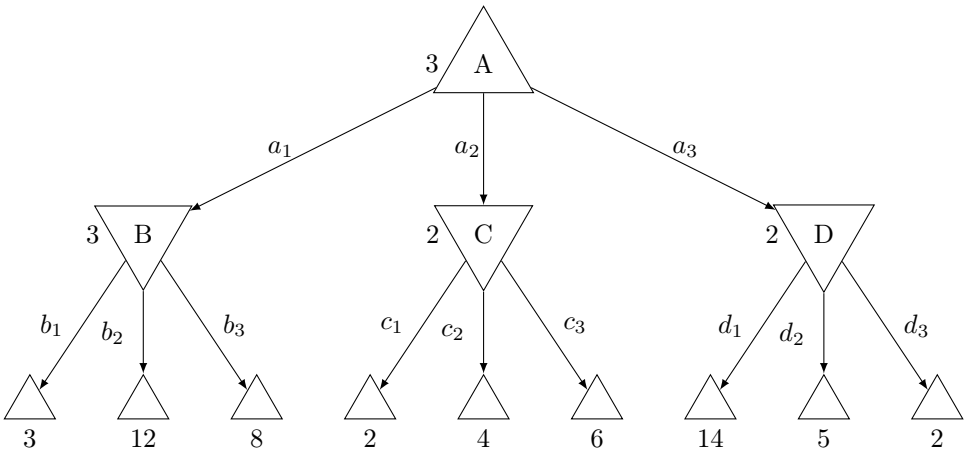


Figura 1.2: *Um jogo simples em duas rodadas (e três turnos). Os vértices* Δ *são vértices* MAX*, em que a vez é do jogador* MAX*, e os vértices* ∇ *são vértices* MIN*. Os vértices terminais mostram o pagamento para* MAX*, enquanto os vértices restantes mostram seus valores minimax. A melhor jogada para* MAX *na raiz é* a_1*, pois leva ao estado B com valor* $3 = max\{3, 2, 2\}$*. A resposta ótima de* MIN *é* b_1*, pois leva ao estado valor* $3 = min\{3, 12, 8\}$*.*

1.7 Algoritmo Minimax

O **algoritmo minimax** computa a decisão minimax a partir do estado de jogo atual. Ele realiza o cálculo dos valores minimax recursivamente de cima para baixo: a recursão procede até atingir os estados terminais, e então os valores são propagados para cima conforme a recursão retorna. O algoritmo realiza uma busca em profundidade completa da árvore de jogo. Assim, se a profundidade máxima da árvore é m e há b jogadas permitidas em cada vértice, a complexidade do algoritmo minimax é $O(b^m)$.

Algorithm 1 MINIMAX-DECISION(*state*) **returns** *an action*

return arg $\max_{a \in \text{ACTIONS(S)}}$ MIN-VALUE(RESULT(*state, a*))

Algorithm 2 MAX-VALUE(*state*) **returns** *a utility value*

if TERMINAL-TEST(*state*) **then**
  return UTILITY(*state*)
$v \leftarrow -\infty$
for all $a \in$ ACTIONS(*state*) **do**
  $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(*state, a*)))
return v

Algorithm 3 MIN-VALUE(*state*) **returns** *a utility value*

if TERMINAL-TEST(*state*) **then**
  return UTILITY(*state*)
$v \leftarrow \infty$
for all $a \in$ ACTIONS(*state*) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(*state, a*)))
return v

Capítulo 2

Jogos Sequenciais

2.1 Jogos de Stackelberg

Em jogos simultâneos, cada jogador escolhe sua ação sem conhecimento das ações escolhidas pelos demais jogadores. Diferentemente de jogos simultâneos, em jogos sequenciais, um jogador escolhe sua ação antes de que os demais escolham as suas. Os agentes em jogos sequenciais recebem nomes diferentes dependendo do domínio de sua aplicação. Quando os agentes são econômicos, como empresas, o primeiro jogador a agir é o líder e os demais são seguidores. Nesse contexto, no qual os jogos de Stackelberg foram inicialmente introduzidos na década de 1930 [18, 20], o líder é tipicamente uma empresa que tem domínio de mercado e as demais empresas seguidoras agem de acordo com a ação do líder. Quando os agentes são de um universo de segurança, um defende possíveis alvos, como pessoas, propriedades, florestas, e os demais procuram tomar ou destruir esses alvos. Assim, a primeira jogadora é chamada de defensora, ou defesa, e os demais de atacantes.

Dessa forma, são jogos em que um líder define uma estratégia e aloca seus recursos de acordo com ela e, na sequência, um seguidor, seu adversário realiza sua jogada com plena ciência da estratégia do líder, agindo racionalmente buscando maximizar seu pagamento. Formalmente, a estratégia do seguidor em um jogo de Stackelberg é uma função que seleciona uma estratégia em resposta à estratégia do líder.

Na forma normal, temos que um jogo de Stackelberg tem dois jogadores, o líder, L, e o seguidor, F, cada um com seu conjunto de estratégias puras, também chamadas de ações, $s_L \in S_L$ e $s_F \in S_F$, e estratégias mistas $\sigma_L \in \Delta S_L$ e $\sigma_F \in \Delta S_F$. A particularidade deste tipo de jogo é que a estratégia do seguidor segue uma função $f : \Delta S_L \to \Delta S_F$ que seleciona uma estratégia para si a partir da estratégia do líder. Para simplicidade de notação, denotaremos por σ_{F_L} a aplicação da função f na estratégia σ_L.

A seguir, iremos ilustrar a vantagem que decorre do fato de que um dos jogadores tem a capacidade de se comprometer com uma estratégia e comunicá-la para os outros jogadores. Caso nenhum dos jogadores possua a vantagem de se comprometer com uma estratégia e os dois façam suas jogadas simultaneamente, conforme exploramos no capítulo anterior, o jogador 1, representado nas linhas, tem uma estratégia estritamente dominante: independentemente da escolha do jogador 2, jogar U sempre lhe dará um pagamento maior do que jogar D. Ao perceber que o jogador 1 sempre se beneficiará de jogar U, o jogador 2 irá preferir jogar L e receber um pagamento de 1, ao invés de jogar R e receber 0. Assim, (U, L) será a solução do jogo, com utilidade de (1, 1) para os jogadores.

	L	R
U	1, 1	3, 0
D	0, 0	2, 1

Tabela 2.1: *Matriz de pagamento para um jogo ilustrativo em forma normal. Exemplo retirado de [2]*

Suponha, porém, que o jogador 1 tem a habilidade de se comprometer com uma estratégia pura e informar ao jogador 2 sua escolha, antes de o jogador 2 realizar sua própria jogada. Neste caso, o jogador 1 passará a ser o líder, enquanto o jogador 2 passará a ser o seguidor. Essa situação, para o mesmo jogo apresentado acima, é apresentada na forma extensiva na figura 2.1. Caso o líder se comprometa com a jogada U, o resultado será semelhante ao do jogo anterior. Mas, ao se comprometer com D, o jogador 2 perceberá que é vantajoso para si jogar R. Neste caso, a utilidade será de (2, 1) para os jogadores, e não (1, 1).

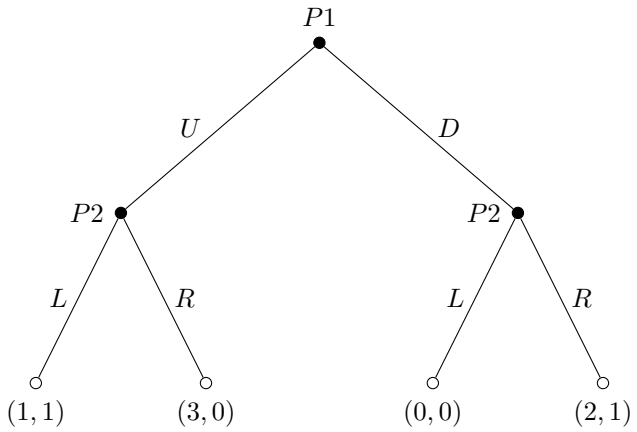


Figura 2.1: *Jogo da tabela 2.1 em forma extensiva*

2.2 Equilíbrio de Stackelberg

O equilíbrio de Stackelberg é um refinamento do equilíbrio de Nash. Ele pode ser entendido como uma forma de equilíbrio perfeito de um subjogo, em que cada jogador escolhe a resposta ótima em qualquer subjogo do jogo original, com um subjogo correspondendo a sequências parciais de ações [20]. No entanto, a estratégia de resposta ótima não garante uma solução única para jogos de Stackelberg, uma vez que é possível que a escolha entre um conjunto de estratégias seja indiferente para o seguidor, ou seja, a escolha entre elas resulte em um pagamento semelhante para ele. Disso decorre a diferença entre um equilíbrio forte, que assume que o seguidor irá escolher a melhor estratégia para o líder quando em face de estratégias indiferentes para si próprio, e um equilíbrio fraco, que assume que o seguidor irá escolher a pior para o líder. Todo jogo de Stackelberg possui um equilíbrio forte, mas nem todo possui um equilíbrio fraco.

Definição 2.1 *Um par de estratégias $(x, f(x))$ forma um equilíbrio forte de Stackelberg se satisfaz o seguinte:*

1. *O líder joga uma resposta ótima: $U_L(x, f(x)) \geq U_L(x', f(x'))$, para todas as estratégias x' do líder.*

2. *O seguidor joga uma resposta ótima:* $U_F(x, f(x)) \geq U_F(x, y')$*, para todas as estratégias* y' *do seguidor.*

3. *O seguidor quebra empates de forma ótima para o líder:* $U_L(x, f(x)) \geq U_L(x, y')$*, para todas as respostas ótimas* y' *do seguidor.*

2.3 Minimax e Stackelberg

A partir da matriz de um jogo de Stackelberg de dois jogadores, é possível utilizar o algoritmo Minimax da seção 1.7 para encontrar uma solução ótima, uma vez que podemos entender a situação como um jogo adversarial em turnos.

Conitzer [3] propõe o pseudo-código aplicável, além de oferecer um exemplo passo a passo da solução. Nele, G é um jogo e $Su(G, s_1)$ representa o subjogo gerado após o primeiro jogador em G utilizar a estratégia $s_1 \in S_1^G$. $\{e\}$ representa um vetor vazio e a função APPEND(s, O) inclui a estratégia s a cada um dos vetores de estatrégia do conjunto O. Iremos escrever $u_1^G(C)$ quando todos os perfis de estratégias contidos no conjunto C dão ao líder o mesmo pagamento no jogo G.

Algorithm 4 SOLVE(G) **returns** *a strategy profile*

if G has 0 players **then**
  return $\{e\}$
$C \leftarrow \emptyset$
for all $s_1 \in S_1^G$ **do**
  $O \leftarrow$ SOLVE$(Su(G, s_i))$
  $O' \leftarrow argmax_{o \in O} u_1^G(s_1, o)$
  if $C = \emptyset$ **or** $u_1^G(s_1, O') = u_1^G(C)$ **then**
    $C \leftarrow C \cup$ APPEND(s_1, O')
  if $u_1^G(s_1, O') > u_1^G(C)$ **then**
    $C \leftarrow$APPEND(s_1, O')
return C

Vamos a um exemplo de como esse algoritmo funciona, considerando um jogo de 3 pessoas, no qual o primeiro jogador escolhe a matriz da esquerda ou da direita, o segundo jogador escolhe a linha e o terceiro jogador escolhe a coluna:

0,1,1	1,1,0	1,0,1
2,1,1	3,0,1	1,1,1
0,0,1	0,0,0	3,3,0

3,3,0	0,2,0	3,0,1
4,4,2	0,0,2	0,0,0
0,5,1	0,0,0	3,0,0

Primeiro eliminamos os resultados que não correspondem aos melhores resultados para o terceiro jogador:

0,1,1		1,0,1
2,1,1	3,0,1	1,1,1
0,0,1		

		3,0,1
4,4,2	0,0,2	
0,5,1		

Agora removemos os resultados em que o terceiro jogador desempataria em favor do segundo jogador, assim como os resultados que não correspondem aos melhores resultados para o segundo jogador:

0,1,1		
2,1,1		1,1,1

0,5,1		

Por fim, removemos os resultados em que o segundo jogador desempataria em favor do primeiro jogador, assim como os resultados que não correspondem aos melhores resultados para o primeiro jogador:

2,1,1		

Assim, o primeiro jogador escolheria a matriz da esquerda, enquanto que o segundo jogador escolheria a linha do meio e o terceiro jogador escolheria a coluna da esquerda.

2.4 Minimax e estratégias mistas

Tentar encontrar a solução de jogos de estratégias mistas representados na forma extensiva é inviável computacionalmente falando, pois eles podem adquirir tamanho infinito [2]: uma vez que um jogador tem a possibilidade de seguir uma estratégia mista, há um continuum de jogadas que podem ser realizadas, gerando infinitos subjogos. Isso ocorre pois decidir uma estratégia mista não é o mesmo que aleatorizar a decisão sobre qual estratégia pura seguir [2]. No caso dos jogos de Stackelberg, é simples visualizar a diferença: caso o líder estivesse escolhendo de maneira aleatória qual estratégia assumir, guiado por determinada distribuição de probabilidade, o seguidor veria apenas a realização do processo aleatório, ou seja, uma estratégia pura selecionada, antes de agir.

Uma vez que a árvore de jogo tem infinitos vértices, ela não pode ser representada explicitamente. Uma outra possibilidade seria discretizar o espaço das estratégias mistas, escolhendo apenas um subconjunto finito delas para computar uma aproximação. Porém, mesmo com alguns artifícios para reduzir o espaço de busca, como a **poda alfa-beta** [16], essa abordagem, além de não representar perfeitamente o jogo, ainda segue sendo computacionalmente ineficiente. Para qualquer N, caso tomássemos apenas as distribuições que utilizam probabilidades múltiplas de $\frac{1}{N}$, ainda teríamos $\binom{N+|S_i|-1}{|S_i|-1}$ distribuições – um valor exponencial com relação às estratégias puras do líder, de forma que seria impossível utilizar essa abordagem e ainda chegar a um algoritmo em tempo polinomial [2].

Dessa forma, um algoritmo para computar a estratégia mista ótima para o líder deverá operar com uma representação diferente do jogo: a forma normal. Uma vez que a computação da estratégia mista em um jogo de Stackelberg é uma generalização da computação de uma estratégia maximin em um jogo de soma zero de dois jogadores [2], podemos utilizar o teorema minimax (1.5) [11] para calcular a estratégia mista ótima para o líder por meio de otimização linear.

O algoritmo utiliza uma abordagem de divisão e conquista da seguinte maneira: para cada estratégia pura $t* \in T$ do seguidor, calculamos a resposta para a seguinte pergunta:

> Qual é a maior utilidade que o líder pode obter, sob a restrição de que o líder joga uma estratégia mista σ_1 para a qual $t*$ é uma resposta ótima?

Esta pergunta, formulada como um programa linear que pode ser resolvido em tempo polinomial, será:

$$\begin{aligned} \text{maximizar} \quad & \sum_{s \in S} p_s u_l(s,t) \\ \text{sujeito a} \quad & \forall t^{'} \in T, \sum_{s \in S} p_s u_f(s,t) \geq \sum_{s \in S} p_s u_f(s,t^{'}) \\ & \sum_{s \in S} p_s = 1 \end{aligned}$$

em que S representa o conjunto de estratégias puras s do líder, T representa o conjunto de estratégias puras t do seguidor, p_s a probabilidade de escolha de uma estratégia pura do líder e $u_f(s,t)$ a função de pagamento para a escolha das estratégias puras s e t. A restrição $\sum_{s \in S} p_s = 1$ garante que o resultado obtido será uma distribuição de probabilidade válida.

Para exemplificar, consideremos o seguinte sistema linear:

$$\begin{aligned} \text{maximizar} \quad & 2x_1 + x_2 \\ \text{sujeito a} \quad & x_1 + x_2 = 1 \\ & 5x_1 + 2x_2 \leq 3 \\ & 7x_1 - 2x_2 \leq 2 \end{aligned}$$

A solução ótima desse sistema é $x_1 = 1/3$ e $x_2 = 2/3$. Ao transformar esse sistema em um jogo com estratégias mistas, em que o líder escolhe a linha e o seguidor escolhe a coluna, obtemos o seguinte:

2, 0	0, 2	0, 5
1, 0	0, -1	0, -4

De fato, a estratégia ótima para o líder consiste em escolher a linha de cima com probabilidade $1/3$ e escolher a linha de baixo com probabilidade $2/3$.

2.5 Jogos de segurança

Em uma evolução mais recente, os jogos de Stackelberg passaram a ser estudados no domínio da segurança [20]. Um jogo de segurança, no contexto deste texto, é uma situação de conflito na qual há uma entidade defensora, responsável pela proteção de bens, como aeroportos, trens, florestas e outros tipos de infraestrutura, em face de um adversário que visa atacá-los. A defesa possui recursos de segurança, como guardas, viaturas, etc., que deve alocar para a proteção dos bens, embora tais recursos sejam em menor número do que os bens a serem protegidos. Por sua vez, o adversário é capaz de monitorar a defesa empregada, sendo capaz de explorar os padrões existentes e usá-los em sua vantagem.

Assim, nessas situações, o papel do líder é representado pela defensora, ou defesa, e o do seguidor, pelo atacante, enquanto as estratégias puras, também chamadas de ações, do líder representam a alocação de seus recursos de segurança para a defesa de determinado alvo, e as do atacante, um ataque a determinado alvo.

O exemplo a seguir tangibiliza a aplicação de um jogo de Stackelberg para o domínio da segurança: Considere um aeroporto com dois terminais. Suponha que há apenas uma patrulha para proteger os dois terminais do ataque de um adversário. O Terminal 1 é considerado mais importante por ser internacional, enquanto o Terminal 2 é doméstico. De posse da informação de que o Terminal 1 é mais importante, a polícia poderia decidir sempre o proteger. No entanto, um atacante inteligente, após um período de monitoramento, perceberia isso e atacaria o Terminal 2. Como não há policiais lá, o ataque é bem sucedido. Por outro lado, a polícia poderia assumir uma estratégia mista, passando 60% dos dias no Terminal 1

e 40% dos dias no Terminal 2, aleatoriamente. O atacante saberia apenas que o Terminal 1 é monitorado 60% dos dias, mas não quais deles, aumentando sua incerteza e, consequentemente, o pagamento esperado da polícia. Ela poderia, ainda, passar 70%, ou 50% dos seus dias no Terminal 1. Assim, a solução do jogo é determinar a estratégia mista ótima para o líder, ou, em termos práticos, definir as ações da defesa em termos de que alvos proteger.

Capítulo 3

Jogos Bayesianos

Nos exemplos anteriores, tratávamos sempre em jogos da forma normal que assumem que todos os agentes têm conhecimento dos pagamentos, e portanto, preferências, dos outros agentes. No entanto, em casos reais, especialmente no domínio da segurança, isso não costuma ser verdade e costuma-se haver informações privadas. Os jogos bayesianos são uma modelagem que responde a essa incerteza sobre as preferências de cada jogador. Neste capítulo, detalharemos os conceitos acerca dos jogos bayesianos e estudaremos alguns algoritmos que utilizam essa modelagem para a computação.

3.1 Informação incompleta

Do ponto de vista da informação, os jogos podem ser divididos em duas classes: jogos de informação completa, e jogos de informação incompleta. A diferença entre eles reside no fato de que, em jogos de informação incompleta, alguns ou todos os jogadores não possuem a informação completa sobre as regras do jogo ou, equivalentemente, sobre sua forma normal. Algumas das informações faltantes podem ser as funções de pagamentos, próprias ou dos seus oponentes, as estratégias disponíveis para cada jogador, ou mesmo quanta informação seus oponentes têm à sua disposição. Aqui, é importante ressaltar a diferença entre jogos de informação completa/incompleta e de informação perfeita/imperfeita. A segunda categorização diz respeito ao conhecimento do histórico de jogadas.

Em um jogo de informação incompleta, a escolha de estratégias depende das **expectativas ou crenças dos jogadores** a respeito das informações faltantes, o que muitas vezes resulta em uma derivação infinita das expectativas. Como exemplo, consideremos um jogo de dois jogadores em que eles não sabem as funções de pagamento um do outro. A estratégia do jogador 1 dependerá da sua crença a respeito da função de pagamento do jogador 2, ou sua *crença de primeira ordem*. Mas sua estratégia também dependerá da sua crença a respeito da crença de primeira ordem do jogador 2 a respeito da sua própria (jogador 1) função de pagamento, ou sua *crença de segunda ordem*. Similarmente, a *crença de terceira ordem* será o que o jogador 1 acredita que seja a crença de segunda ordem do jogador 2, e assim sucessivamente [7].

A **abordagem bayesiana** consiste em modelar as crenças dos jogadores por meio de distribuições de probabilidades subjetivas. Uma **distribuição de probabilidade subjetiva** é definida em termos do comportamento de escolha do jogador, em oposição a uma distribuição de probabilidade objetiva que é baseada nas frequências observadas dos eventos relevantes a longo prazo. Em outras palaras, a probabilidade subjetiva refere-se à probabilidade de algo acontecer com base na própria experiência ou julgamento pessoal de um indivíduo. Uma probabilidade subjetiva não se baseia em dados de mercado ou informações

históricas e difere de pessoa para pessoa, trata-se de uma probabilidade condicional que depende da visão/passado/crença do jogador. Por conta dessa abordagem, jogos de informação incompleta também são chamados de **jogos bayesianos**.

Mantendo o exemplo anterior, a crença de primeira ordem do jogador 1 terá a forma de uma distribuição de probabilidade subjetiva $P_1^1(U_2)$ sobre todas as possíveis funções de pagamento U_2 do jogador 2, e sua crença de segunda ordem terá a forma $P_1^2(P_2^1)$, uma distribuição de probabilidade subjetiva sobre todas as possíveis crenças de primeira ordem P_2^1 que o jogador 2 pode ter sobre a função de pagamento do jogador 1.

Evidentemente, a abordagem bayesiana não é uma forma prática para modelar o problema, especialmente em jogos com mais de dois jogadores. Harsanyi [7] propõe uma alternativa, que consiste em transformar jogos de informação incompleta em jogos de informação completa e imperfeita que sejam equivalentes ao jogo original.

Em seu trabalho, Harsanyi trata jogos incompletos em que há uma rede complexa de incertezas e crenças envolvendo as funções de pagamentos dos jogadores. Descreveremos tais jogos de uma perspectiva simplista que esperamos que seja suficientemente precisa para os nossos propósitos.

3.2 Hipótese bayesiana

Ao analisar jogos de informação incompleta, Harsanyi [7] assume que, ao enfrentar uma situação de informação incompleta, todos os jogadores utilizarão uma abordagem bayesiana, ou seja, assumirão uma distribuição de probabilidade subjetiva P_i para todas as variáveis desconhecidas a eles. A partir disso, eles tentarão maximizar seu pagamento x_i em termos da distribuição de probabilidade P_i. Essa premissa é chamada de **hipótese bayesiana**.

Harsanyi [7] demonstra que, apesar de haver diversas categorias em que é possível haver informação incompleta em um jogo, como recursos disponíveis para si e para o adversário, função de pagamento própria e do adversário, mapeamento entre estratégias e pagamentos, entre outros, é possível reduzir todos os tipos de informação incompleta a falta de conhecimento sobre a função de pagamento. Nesses cenários, há possíveis crenças sobre quanto um jogador j acredita que o jogador k conhece da função de pagamento do jogador i.

A redução do jogo de informação incompleta para um jogo de informação completa envolve o uso de eventos aleatórios que ocorrem antes de os jogadores escolherem suas estratégias s_1 e s_2. Estes eventos determinam a função de pagamento para cada um deles e, em parte da literatura subsequente, esses eventos ou jogadas aleatórias que determinam a sequência do jogo mas não são associados aos competidores são metaforicamente chamados de jogadas do *jogador natureza*, que representa a concretização desses eventos aleatórios. Assume-se que ambos os jogadores conhecem a distribuição de probabilidade conjunta destes eventos, mas têm acesso à realização dessa probabilidade apenas para os eventos que dizem respeito a si próprios. Dessa forma, o jogo é dito de informação completa, mas imperfeita, uma vez que os jogadores não possuem conhecimento total do histórico de eventos ocorridos durante o jogo. Harsanyi sugere uma interpretação alternativa, mais intuitiva: ao invés de assumir que certos atributos importantes dos jogadores são determinados por eventos aleatórios hipotéticos no início do jogo, podemos assumir que os próprios jogadores são selecionados aleatoriamente de uma população hipotética que contém um misto de indivíduos de **tipos** diferentes, caracterizados por vetores de atributos diferentes.

Os **vetores de atributo** são definidos por Harsanyi como a representação dos atributos físicos, sociais e psicológicos dos jogadores, de forma que sintetizam certos parâmetros cruciais da função de pagamento do jogador correspondente. Também podem ser chamados de **vetores de informação**, uma vez que representam toda a informação disponível para

cada jogador durante o jogo. Eles são representados na forma $c_i = (a_i, b_i)$ para cada jogador i, em que $a_i = (a_{i_1}, a_{i_2}, \ldots, a_{i_n})$ é o vetor que contém todas as informações a respeito das n funções de pagamento $U_1, \ldots, U_n$ em um jogo de n jogadores, enquanto b_i será o vetor de parâmetros da distribuição de probabilidade subjetiva P_i. De maneira similar, $c = (a, b)$ pode ser escrito como $c = (c_1, c_2, \ldots, c_n) = ((a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n))$ e $c_{-i} = (a^i, b^i)$ como o vetor derivado de c que omite c_i, ou seja, $c_{-i} = (c_1, c_2, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) = ((a_1, b_1), (a_2, b_2), \ldots, (a_{i-1}, b_{i-1}), (a_{i+1}, b_{i+1}), \ldots, (a_n, b_n))$.

Seja G um jogo de informação incompleta com um conjunto $\{1, 2, \ldots, n\}$ de jogadores. A partir deste ponto fixaremos um jogador qualquer j e analisaremos G da perspectiva de j. Mais adiante denotaremos por G_j o subjogo determinado por essa perspectiva. Pretendemos que G seja o jogo resultante da composição das perspectivas que seus jogadores tem sobre o jogo; em símbolos, $G = (G_1, G_2, \ldots, G_n)$.

Como temos feito, para $i = 1, 2, \ldots, n$, S_i e U_i denotarão, respectivamente, o conjunto das estratégias e função de pagamento do jogador i. Diferentemente do que tínhamos feito até agora, cada função de pagamento U_i, além de continuar a depender das estratégias adotadas pelos jogadores, dependerá ainda dos parâmetros que o jogador j acredita que são de conhecimento dos demais jogadores. Se $a_{k,i}$ é o subvetor de c_k em C_k com os parâmetros que j acredita que k conhece sobre a função de pagamento U_i, teremos que o pagamento recebido por i é

$$U_i(s_1, s_2, \ldots, s_n, a_{1,i}, a_{2,i}, \ldots, a_{n,i}).$$

No jogo G_j suporemos que para $i = 1, 2, \ldots, n$ cada um dos jogadores tem conhecimento completo sobre uma função de distribuição de probabilidades R_{-i} sobre $C_1 \times C_2 \times \cdots \times C_{i-1} \times C_{i+1} \times \cdots C_n$, em que C_i é o conjunto de valores que c_i pode assumir. Como c_i é um vetor, evidentemente, cada um de seus componentes tem o seu próprio conjunto de possíveis valores (=*range space*). Dessa forma, cada um dos jogadores conhece a distribuição de probabilidade sobre os vetores de informações de todos os outros jogadores sob sua perspectiva.

Finalmente, se $\mathcal{S} = (S_1, S_2, \ldots, S_n)$, $\mathcal{C} = (C_1, C_2, \ldots, C_n)$, $\mathcal{U} = (U_1, U_2, \ldots, U_n)$, $\mathcal{R} = (R_{-1}, R_{-2}, \ldots, R_{-n})$, então dizemos que $G_j = (\mathcal{S}, \mathcal{C}, \mathcal{U}, \mathcal{R})$ é jogo de informação incompleta sob a perspectiva de um jogador j.

3.3 Transformação de Harsanyi

Harsanyi [7] define o jogo naturalmente análogo do jogo G, de informação incompleta, descrito acima como o jogo G^*, de informação completa, com as mesmas funções de pagamento $U_1, \ldots, U_n$ e o mesmo espaço de estratégias S_i. No entanto, em G^*, os vetores c_i serão reinterpretados como vetores aleatórios (que determinam jogadas aleatórias) com uma distribuição de probabilidade conjunta objetiva $R^* = R^*(c_1, \ldots, c_n) = R^*(c)$ conhecida por todos os jogadores. Para que G^* seja o mais similar possível a G, cada vetor c_i terá seus valores originados do mesmo espaço C_i do jogo G e, além disso, quando o jogador i escolhe sua estratégia s_i, ele saberá apenas o valor do seu próprio vetor aleatório c_i, mas não o do restante dos vetores $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n$ pertencente ao restante dos jogadores. Note que, portanto, G é o jogo de informação incompleta do qual tratávamos na seção anterior, enquanto G^* é o jogo bayesiano-equivalente a G e, portanto, um, jogo de informação completa.

Dizemos que um jogo de informação completa está na forma normal se:

1. as funções pagamento $U_1, \ldots, U_n$ de G^* estão na forma
   $x_i = U_i(s_1, \ldots, s_n, c) = U_i(s_1, \ldots, s_n, c_1, \ldots, c_n)$

2. os vetores $c_1, \ldots, c_n$ são vetores aleatórios com uma distribuição de probablidade conjunta R^* conhecida por todos os jogadores

3. cada jogador i tem conhecimento apenas de seu próprio vetor c_i e não tem conhecimento dos vetores $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n$ pertencente ao restante dos jogadores.

Formalmente, definimos:

Definição 3.1 *Um jogo de informação completa em forma normal é representado por um conjunto ordenado tal que* $G^* = (S_1, \ldots, S_n, C_i, \ldots, C_n, U_1, \ldots, U_n, R^*)$.

Dessa forma, G^* difere de G apenas pela substituição da n-tupla $(R_1, \ldots R_n)$ por uma única distribuição de probabilidade conjunta objetiva R^*. A partir das definições de G e G^*, definimos a relação de equivalência entre os jogos conforme abaixo:

Definição 3.2 (Jogos Bayesianos-Equivalentes) *Seja G um jogo de informação incompleta, do ponto de vista do jogador j, e G^* um jogo de informação completa, ambos na forma normal. Dizemos que G e G^* são bayesianos-equivalentes se as seguintes condições são satisfeitas:*

- *Os dois jogos possuem o mesmo espaço de estratégias $S_1, \ldots, S_n$*
- *Os dois jogos possuem as mesmas funções de pagamento $U_1, \ldots, U_n$*
- *A distribuição subjetiva de probabilidade R_i de cada jogador i em G deve satisfazer a relação $R^*(c) = R^*(c_i, c') = R_i(c'|c_i) \int_{c'} d_{c'} R^*(c_i, c')$, em que $R^*(c) = R^*(c_i|c')$ é a distribuição de probabilidade do jogo G^*.*

O jogo bayesiano-equivalente de informação completa é chamado de **jogo bayesiano**. Assim, em um jogo bayesiano, cada jogador i possui um conjunto de ações S_i, um conjunto de tipos θ_i, sendo um tipo definido conforme na seção anterior, com uma distribuição de probabilidade associada $\pi_i : \theta_i \to [0, 1]$ e, para cada tipo θ_i, uma função de pagamento $u_i^{\theta_i} : S_1 \times S_2 \times \ldots \times S_n \to \mathbb{R}$.

A transformação de um jogo de informação incompleta em um jogo de informação completa bayesiano-equivalente é hoje conhecida como **transformação de Harsanyi**, em homenagem ao autor [7].

Embora conceitualmente complexo, a implementação da transformação é razoavelmente simples, como ilustraremos a seguir, com o problema conhecido como "Dilema do Xerife". Suponha uma situação em que um xerife enfrenta um suspeito armado, e ambos precisam decidir simultaneamente se irão atirar ou não em seu oponente. O suspeito pode ser do tipo *criminoso* ou do tipo *civil*, enquanto o xerife tem apenas um tipo. O Xerife não sabe o tipo do suspeito antes de tomar a decisão de atirar ou não, o que faz com o que o dilema seja um jogo bayesiano. O Xerife sabe que há uma probabilidade de $\frac{2}{3}$ de que o suspeito seja criminoso e de $\frac{1}{3}$ de que seja civil, e o suspeito também está ciente dessa probabilidade – este é o equivalente de R^* na notação formal.

Para o suspeito civil, temos a seguinte matriz de pagamento, com o xerife nas colunas e o suspeito nas linhas:

	Atira	Não Atira
Atira	-3, -1	-1, -2
Não Atira	-2, -1	0, 0

Já para o suspeito criminoso, temos a matriz de pagamento a seguir, novamente com o xerife nas colunas e o suspeito nas linhas:

	Atira	Não Atira
Atira	0, 0	2, -2
Não Atira	-2, -1	-1, 1

Ao incorporar o jogador natureza, que seleciona o adversário do tipo *criminoso* com probabilidade $\frac{2}{3}$ e *civil* com probabilidade $\frac{1}{3}$, conseguimos combinar as duas matrizes de pagamento em uma só e tratar o jogo como um jogo de informação completa e imperfeita.

	Atira	Não Atira
Atira	$-1, -1/3$	$1, -2$
Não Atira	$-2, -1$	$-2/3, 2/3$

3.4 Jogos de Stackelberg bayesianos

Em um jogo de Stackelberg, como já vimos, teremos dois agentes: o líder e o seguidor, que no domínio da segurança assumem os papéis de defesa - ou defensora - e atacante, respectivamente. Assim, em um jogo de Stackelberg bayesiano básico, mantemos esses papéis e assumimos, de maneira razoável que há apenas um tipo de defesa, como a polícia, enquanto há mais de um tipo de atacante, como diferentes grupos terroristas, cada um com seus interesses. A defesa não sabe qual tipo está enfrentando.

É importante ressaltar que a transformação de Harsanyi apresentada anteriormente e ilustrada na forma normal impõe um desafio computacional para jogos sequenciais, pois embora a forma normal após a transformação contenha o mesmo conjunto de ações para o líder que no jogo original, o conjunto das ações possíveis do seguidor será o produto cartesiano do conjunto das ações possíveis para cada tipo de seguidor. Assim, o número de linhas da matriz se mantém estável, mas o número de colunas aumenta exponencialmente, já que a todas as combinações de sequências de jogadas de cada tipo de seguidor serão consideradas como uma ação possível.

Capítulo 4

Resolução de Jogos Bayesianos

Como foi mostrado por Conitzer e Sandholm [3], encontrar uma estratégia mista ótima para um jogo com pelo menos três jogadores na forma normal é NP-difícil. Os autores ainda mostraram que, para jogos sequenciais bayesianos, bastam dois jogadores para tornar a tarefa de encontrar uma estratégia ótima NP-difícil. Esses fatos são chamados de intratabilidade, já que mostram que nessas situações não há, ou não se acredita que haja, métodos eficientes para encontrar estratégias ótimas. Neste capítulo, apresentaremos alguns algoritmos que buscam resolver jogos de segurança, ou seja, encontrar a estratégia mista ótima para a defesa, considerando que o atacante pode ter consciência dela ao escolher sua estratégia.

4.1 DOBSS

O *Decomposed Optimal Bayesian Stackelberg Solver* (DOBSS) [13] é o algoritmo do sistema ARMOR, que é utilizado no processo de segurança do Aeroporto Internacional de Los Angeles (LAX). Como seu nome sugere, o DOBSS utiliza-se de um esquema de decomposição ao tirar vantagem da independência de cada atacante. A chave dessa decomposição encontra-se no fato de que avaliar a estratégia da defesa na forma normal bayesiana-equivalente, ou seja, após a transformação de Harsanyi, é equivalente a avaliar a mesma estratégia individualmente contra cada atacante.

É válido mencionarmos três vantagens desse algoritmo:

- Não há necessidade de transformar este jogo em forma normal, dispensando o uso da transformação de Harsanyi e usufruindo da independência entre os diferentes tipos de atacantes;
- Este algoritmo utiliza apenas um *Mixed-Integer Linear Program* (MILP) ao invés de um conjunto de sistemas lineares, obtendo uma melhora no quesito performance;
- O DOBSS procura diretamente pela estratégia ótima da defesa, ao invés do equilíbrio de Nash (ou Bayes-Nash), explorando a vantagem de ser a líder ao permitir encontrar estratégias para si de alta recompensa e que não levam ao equilíbrio.

Os autores iniciam apresentando o problema em um *Mixed-Integer Quadratic Program* (MIQP), mais intuitivo, para então apresentar o problema MILP equivalente.

Nele, as ações da defesa são representadas explicitamente, bem como as ações para cada tipos de atacante e suas condições de otimalidade.

Seja L o conjunto de tipos de atacantes e Q o conjunto das suas estratégias puras. Seja ainda X o conjunto das estratégias puras da defesa. Para cada tipo de atacante l em L

denotamos por q^l o vetor indicador das suas estratégias e denotamos por x o vetor das estratégias mistas da defesa. Assim, q^l é um vetor indexado por Q. Denotaremos por x um vetor indexado por X. As matrizes de pagamento da defesa associada a cada um dos tipos de atacante l a matriz de pagamento de cada um dos tipos de atacante l serão D^l e A^l, respectivamente. A probabilidade de um atacante do tipo l ser selecionado será p^l. Finalmente, suporemos que M é um número suficientemente grande. A defesa deve resolver o seguinte problema:

$$
\begin{aligned}
\max_{x,q,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l D^l_{ij} x_i q^l_j \\
\text{sujeito a} \quad & \sum_{i\in X} x_i = 1 && (1)\\
& \sum_{j\in Q} q^l_j = 1 \\
& 0 \le (a^l - \sum_{i\in X} A^l_{ij} x_i \le (1-q^l_j)M && (2)\\
& x \ge 0 && (3)\\
& q^l_j \in \{0,1\} && (4)\\
& a \in \mathbb{R}^{\mathbb{L}} && (5)
\end{aligned}
$$

Em palavras, $\sum x_i = 1, x \ge 0$ diz que x é um vetor de probabilidades, e $\sum q^l_j, q^l_j \in \{0,1\}$ diz que o tipo de atacante l deve escolher uma e só uma estratégia.

O problema maximiza o pagamento esperado para o agente, para um conjunto de ações x da defesa e q^l para cada atacante $l \in L$, considerando a distribuição a priori dos diferentes tipos de atacante p^l. As restrições *(1)* e *(3)* definem o conjunto de soluções viáveis como uma distribuição de probabilidade sobre o conjunto de ações X, enquanto 2 e 5 limitam o vetor de ações q^l do tipo de seguidor l a ser uma distribuição pura sobre o conjunto de estratégias Q.

A restrição *(2)* garante que $q^l_j = 1$ apenas para uma estratégia j que seja ótima para o tipo de atacante l. A desigualdade da esquerda impõe que, dado o vetor da defesa x, a^l é um limite superior para a recompensa do atacante tipo l para qualquer ação. A desigualdade da direita é irrelevante para toda ação em que $q^l_j = 0$, já que M é uma quantidade positiva alta. Quando $q^l_j = 1$, a desigualdade impõe que o pagamento do atacante por essa ação deve ser $\ge a^l$, o que, conjuntamente com a desigualdade da esquerda, mostra que essa será a estratégia ótima para o atacante tipo l.

O programa, apesar de quadrático, pode ser linearizado ao fazer a mudança de variável $z^l_{ij} = x_i q^l_j$, resultando no seguinte problema, que pode ser resolvido por pacotes de programação linear eficientes:

$$
\begin{aligned}
\max_{q,z,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l D^l_{ij} z^l_{ij} \\
\text{sujeito a} \quad & \sum_{i\in X}\sum_{j\in Q} z^l_{ij} = 1 && (1)\\
& \sum_{j\in Q} z^l_{ij} \leq 1 && (2)\\
& q^l_j \leq \sum_{i\in X} z^l_{ij} \leq 1 && (3)\\
& \sum_{j\in Q} q^l_j = 1 && (4)\\
& 0 \leq a^l - \sum_{i\in X} A^l_{ij}(\sum_{h\in Q} z^l_{ih}) \leq (1-q^l_j)M && (5)\\
& \sum_{j\in Q} z^l_{ij} = \sum_{j\in Q} z^l_{ij} && (6)\\
& z^l_{ij} \in [0...1] && (7)\\
& q^l_j \in \{0,1\} && (8)\\
& a \in \mathbb{R} && (9)
\end{aligned}
$$

O DOBSS foi empregado pela primeira vez por meio do projeto ARMOR [14] para o planejamento das patrulhas caninas e *checkpoints* rodoviários no aeroporto de Los Angeles, ilustrados na figura 4.1. Por meio de uma interface interativa, três entradas são inicialmente fornecidas pelo usuário: (i) número de *checkpoints* simultâneos em uma janela de tempo, (ii) duração da janela de tempo, (iii) número de dias de duração da estratégia – i.e., iterações do jogo. Há, ainda, três inputs que funcionam como restrições sobre a solução para acomodar necessidades específicas do usuário: (i) *checkpoint forçado*, que determina um checkpoint em horário e local específico fornecidos pelo usuário (ii) *checkpoint proibido*, que determina a não existência de um checkpoint em horário e local específico e (iii) *pelo menos um checkpoint*, que determina que pelo menos um checkpoint seja estabelecido em um conjunto de janelas de tempo fornecido pelo usuário.

Figura 4.1: *Checkpoints e patrulhas caninas no aeroporto de Los Angeles. Retirado de [9]*

Uma matriz de jogo é construída a partir dessas informações, e matrizes de pagamento são fornecidas por especialistas de segurança com pagamentos pré-especificados para a polícia e os diferentes tipos de adversários. As matrizes são enviadas para a implementação do DOBSS, que escolhe a estratégia mista ótima a ser seguida. Na sequência, o programa gera um conjunto de ações baseado nessa estratégia. Por exemplo, considerando 3 *checkpoints*, A, B e C, e uma única janela de tempo, suponhamos que a estratégia mista ótima seja [0.70, 0.15, 0.15]. O programa irá gerar, então, um cronograma que seleciona o checkpoint a

partir dessa distribuição de probabilidade.

4.2 ERASER

O *Efficient Randomized Allocation of Security Resources* (ERASER-C) tem origem no projeto IRIS [9], com o objetivo de auxiliar no cronograma de alocação de *Federal Air Marshals* americanos. Os *air marshals* são oficiais à paisana que embarcam em voos com origem ou destino nos Estados Unidos treinados para dissuadir potenciais terroristas e prevenir ataques. A particularidade desse domínio é que eles devem ser alocados em dezenas de milhares de voos por dia, levando em conta a variação de lotação de passageiros, rotas aéreas sobre áreas densamente povoadas, além das próprias rotas de ida/volta dos voos e a localização dos *marshals*.

Por representar explicitamente todas as estratégias puras da defesa, o DOBSS depara-se com o problema do crescimento do número de estratégias de defesa descrito na seção anterior. Para mitigar esse efeito, o ERASER utiliza-se do fato de que, em jogos de segurança, o pagamento depende apenas de o alvo estar coberto, ou seja, não importa se, durante um ataque, há patrulhas também em alvos não atacados. Assim, do ponto de vista do pagamento, diversas alocações de patrulhas são idênticas, levando a uma representação mais compacta que evita a enumeração de todas as combinações possíveis. De forma ilustrativa, em um cenário com 3 guardas defendendo 10 alvos, no DOBSS haveriam $\binom{10}{3} = 120$ ações possíveis, enquanto na representação do ERASER e ERASER-C seriam 10 ações, uma para cada alvo, e uma distribuição de probabilidade indicando como os 3 guardas seriam alocados. As tabelas 4.1 e 4.2 retiradas de [20] explicitam a diferença entre as duas representações.

Ação	Alvos Protegidos	Probabilidade
1	1, 2, 3	p_1
2	1, 2, 4	p_2
3	1, 2, 5	p_3
...	...	...
120	8, 9, 10	p_{120}

Tabela 4.1: *Ações possíveis do DOBSS no cenário com 3 guardas e 10 alvos*

Representação compacta para jogos de segurança

Seja $T = \{t_1, \ldots, t_n\}$ um conjunto de alvos sujeitos a serem atacados, correspondentes a estratégias puras do atacante, e $R = \{r_1, \ldots, r_n\}$ o conjunto de recursos da defesa para proteger esses alvos. Assume-se que esses recursos sejam idênticos e que possam ser alocados a qualquer alvo. Associados a cada alvo, há quatro pagamentos que definem os possíveis

Ação	Alvos Protegidos	Probabilidade
1	1	p_1
2	2	p_2
3	3	p_3
...	...	...
10	10	p_10

Tabela 4.2: *Ações possíveis do ERASER-C no cenário com 3 guardas e 10 alvos*

resultados de um ataque, ilustrados na tabela 4.3. Chama-se um ataque a um alvo protegido de **coberto** e a um alvo não protegido de **descoberto**. O pagamento da defesa por um ataque descoberto é denotado $U^u_\Theta(t)$, e por um coberto, $U^c_\Theta(t)$, e, de forma similar, dos atacantes, $U^u_\Psi(t)$ e $U^c_\Psi(t)$. Os pagamentos dependem apenas da identidade do alvo e de se está ou não coberto.

	Coberto	Descoberto
Defesa	5	−20
Ataque	−10	30

Tabela 4.3: *Pagamentos ilustrativos pelo ataque a um determinado alvo*

A estratégia da defesa é então resumida num vetor de cobertura C, que reflete a probabilidade c_t de cada alvo t estar ou não coberto. O vetor de ataque análogo A reflete a probabilidade de cada alvo ser atacado. O pagamento esperado da defesa é dado pela equação (4.1) enquanto o pagamento de um ataque no alvo t, é dado pela equação (4.2). O pagamento esperado do atacante é análogo.

$$U_\Theta(C, A) = \sum_{t \in T} a_t \cdot (c_t \cdot U^c_\Theta(t) + (1 - c_t)U^u_\Theta(t)) \tag{4.1}$$

$$U_\Theta(t, C) = c_t U^c_\Theta(t) + (1 - c_t)U^u_\Theta(t) \tag{4.2}$$

Define-se o conjunto de ataque $\Gamma(C)$ como o conjunto que contém todos os alvos que fornecem o pagamento esperado máximo para o atacante dada a cobertura C.

$$\Gamma(C) = t : U_\Psi(t, C) \geq U_\Psi(t', C) \forall t' \in T \tag{4.3}$$

Num equilíbrio forte de Stackelberg, o atacante escolhe o alvo no conjunto de ataque com o maior pagamento para a defesa.

Resolução do jogo pelo algoritmo ERASER

Uma vez na forma compacta, o algoritmo resolve o seguinte problema de programação linear:

$$\begin{array}{lll}
\max & d & \\
\text{sujeito a} & \sum_{t \in T} a_t = 1 & \\
& \sum_{t \in T} c_t \leq m & \\
& d - U_\Theta(t, C) \leq (1 - a_t) \cdot Z & \forall t \in T \\
& 0 \leq k - U_\Psi(t, C) \leq (1 - a_t) \cdot Z & \forall t \in T \\
& a_t \in \{0, 1\} & \forall t \in T \\
& c_t \in [0, 1] & \forall t \in T
\end{array}$$

Note que as restrições garantem que o vetor de ataque contenha um único alvo com probabilidade 1 e que a cobertura dos alvos seja menor ou igual ao número de recursos disponíveis m. Z é definida como uma constante alta em relação ao valor de pagamento máximo, e o racional do programa, especialmente das desigualdades, é similar ao do DOBSS. Uma solução ótima dada pelo ERASER corresponde a Equilíbrio Forte de Stackelberg do jogo [9].

4.3 ERASER-C

O ERASER-C é uma variante do ERASER que permite restrições de cronograma de patrulhamento e de tipos de recursos, cada um com a capacidade de proteger alvos diferentes, por exemplo, patrulhas de cães farejadores, policiais e cavalaria.

Define-se como $S = s_1, \ldots, sl$ o conjunto de cronogramas permitidos, e como $M : S \times T \rightarrow 0, 1$ a relação entre alvos e cronogramas, que é avaliada em 1 se e somente se t está coberto em s. A estratégia da defesa passa a ser uma alocação de recursos a cronogramas, e não mais a alvos. Define-se $\Omega = \{\omega_1, \ldots, \omega_v\}$ como o conjunto de tipos de recursos disponíveis, sendo que o número disponível de cada tipo é dado pela função $R(\omega)$ e a capacidade de cobertura é dada pela função $Ca : S \times \Omega \rightarrow \{0, 1\}$, que é mapeada a 1 caso o tipo ω seja capaz de cobrir dado cronograma e a 0 em caso contrário.

O programa é semelhante ao visto na seção anterior, mas se utiliza de restrições adicionais:

$$
\begin{array}{lll}
\max & d & \\
\text{sujeito a} & \sum_{t \in T} a_t = 1 & (1) \\
& \sum_{t \in T} c_t \leq m & (2) \\
& d - U_\Theta(t, C) \leq (1 - a_t) \cdot Z \qquad \forall t \in T & (3) \\
& 0 \leq k - U_\Psi(t, C) \leq (1 - a_t) \cdot Z \qquad \forall t \in T & (4) \\
& \sum_{\omega \in \Omega} h_{s,\omega} = q_s \qquad \forall s \in S & (5) \\
& \sum_{s \in S} q_s M(s, t) = c_t \qquad \forall t \in T & (6) \\
& \sum_{s \in S} h_{s,\omega} CA(s, \omega) \leq R(\omega) \qquad \forall \omega \in \Omega & (7) \\
& \sum_{t \in T} c_t \leq m & (8) \\
& h_{s,\omega} \leq Ca(s, \omega) \qquad \forall s, \omega \in S \times \Omega & (9) \\
& c_t \in [0, 1] \qquad \forall t \in T & (10) \\
& q_s \in [0, 1] \qquad \forall s \in S & (11) \\
& h_{s,\omega} \in [0, 1] \qquad \forall s, \omega \in S \times \Omega & (12)
\end{array}
$$

Além desses trabalhos pioneiros, foram implementados o GUARDS [15], o PROTECT [17] e o TRUSTS [23], entre outros, que são variações dos algoritmos acima para domínios específicos que permitiam otimizações para cada situação.

4.4 ARMOR e IRIS

O DOBSS e o ERASER-C fornecem meios de resolver jogos de Stackelberg, tendo sido desenhados para problemas reais. Para sua aplicação no dia a dia, foram criados dois assistentes de software - o ARMOR, que utiliza o DOBSS no back-end para programar as patrulhas do aeroporto de Los Angeles, e o IRIS, que utiliza o ERASER-C no back-end para programar a alocação dos *marshals*, ambos com apoio da biblioteca `glpk`[1], da GNU, para resolução dos programas lineares. A estrutura geral dos aplicativos é descrita na figura 4.2, retirada de [9]. Exploraremos em detalhe essa arquitetura na subseção seguinte.

[1] https://gnu.org/software/glpk

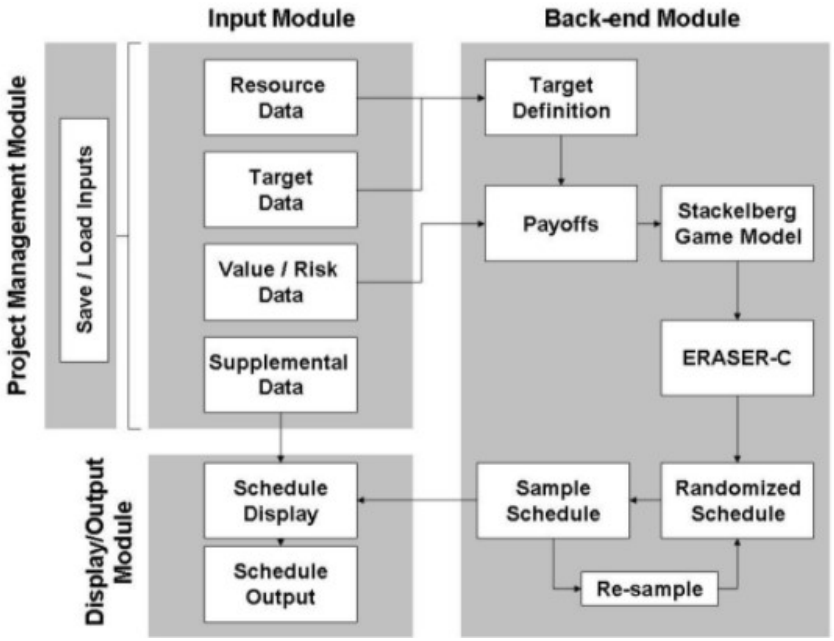


Figura 4.2: *Arquitetura dos aplicativos que implementam o DOBSS e o ERASER-C. Retirado de [9]*

Arquitetura dos programas

Módulo de input

Os programas dependem dos usuários e de experts do setor de segurança para obter o conhecimento específico da área de aplicação para gerar o modelo de jogo. Embora alguns elementos não mudem ao longo do tempo e possam ser embutidos no código, outros mudam com frequência, de forma que precisam ser inseridos pelo usuário. Algumas categorias de input necessários são:

1. Número de recursos disponíveis e suas habilidades
2. Número de alvos
3. Pagamento para cada alvo
4. Dados suplementares para melhorar a experiência do usuário, como nomes dos guardas/cães, nome das localizações, etc.

A quantidade de informação inputada pelo usuário varia entre os dois programas. Por exemplo, no ARMOR, o conjunto de alvos é embutido no código, uma vez que o número de terminais no aeroporto não muda ao longo do tempo. Já no IRIS, é necessário prover a informação, uma vez que o número de voos muda todos os dias. Os autores forneceram as figuras 4.3 e 4.4, que ilustram a interface do ARMOR para as patrulhas caninas e para os *checkpoints* rodoviários, respectivamente.

Uma vez preenchidos os inputs, o programa gera um jogo diferente para cada janela de tempo a cada dia. O número de recursos de defesa é o número de *checkpoints* /patrulhas fornecidos pelo usuário e o número de alvos é o número de vias que chegam ao aeroporto, no caso dos *checkpoints* , e o número de terminais no caso de patrulhas caninas. O pagamento para cada alvo depende de uma variedade de condições, como a quantidade de passageiros, custo da infraestrutura, etc. Assim, experts da área de segurança e antiterrorismo criaram

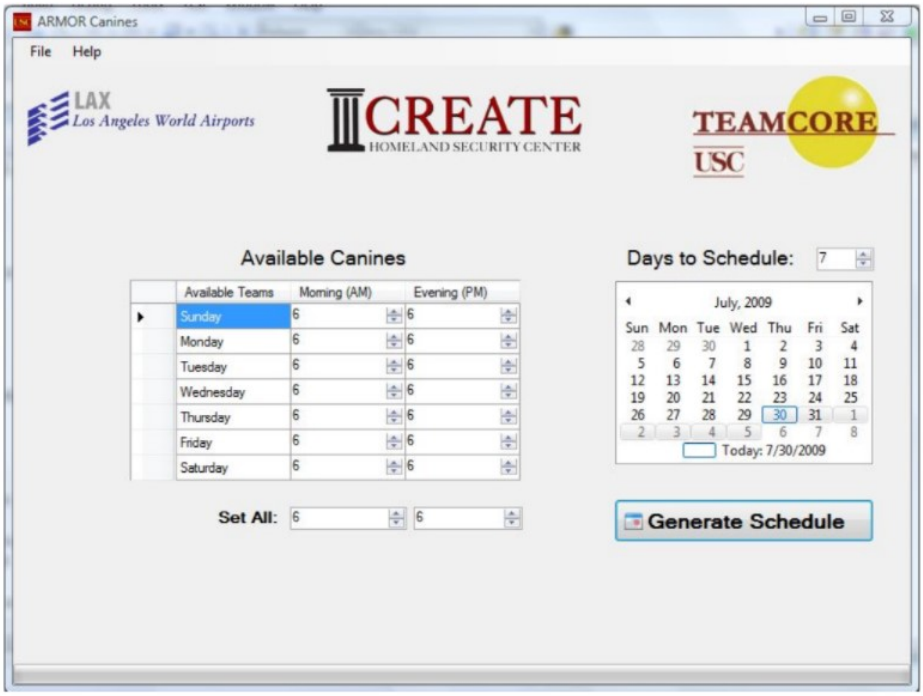


Figura 4.3: *Interface gráfica para programação de patrulha canina do ARMOR. Retirado de [9]*

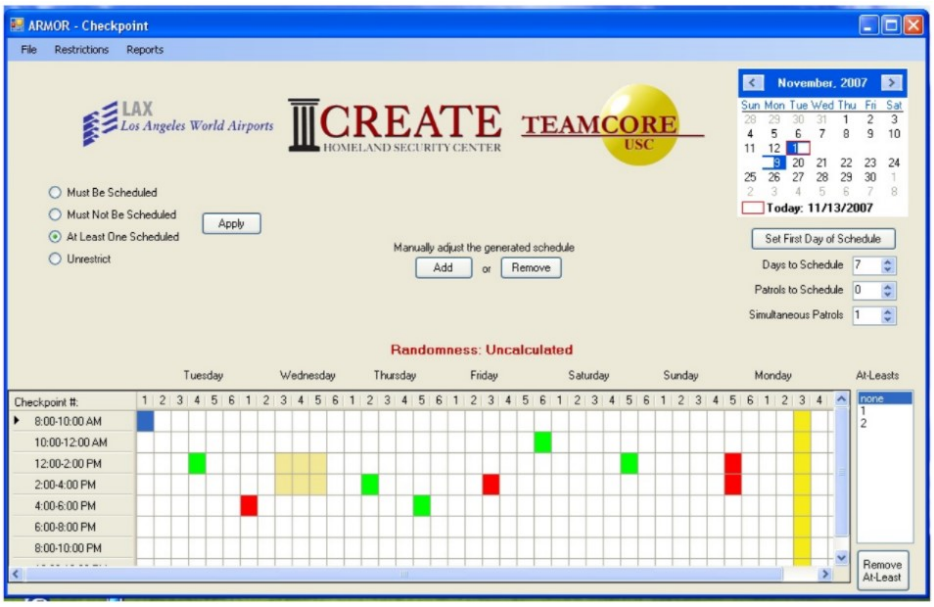


Figura 4.4: *Interface gráfica para programação de checkpoint rodoviário do ARMOR. Retirado de [9]*

uma fórmula para calcular automaticamente os pagamentos considerando todas as diferentes combinações dessas condições. Dessa forma, não é necessário que o usuário cadastre esses pagamentos, embora possa fazê-lo em situações com informações específicas – como quando os serviços de inteligência recebem informação de um ataque específico.

Para o IRIS, o usuário deve fornecer mais informações; no entanto, a abordagem é similar à descrita anteriormente.

Criação do jogo e resolução

Esse módulo constrói uma instância específica de um jogo de Stackelberg Bayesiano, com base nos dados fornecidos pelos experts e pelos usuários através da interface guiada. Uma vez que um jogo explícito é criado, ele é passado como input para o DOBSS ou para o ERASER-C, e resolvido através da biblioteca `glpk`. Os programas retornam uma estratégia mista ótima para a defesa, ou seja, uma distribuição de probabilidade sobre as ações da defesa. Com base nessa probabilidade, é gerado um cronograma que mostra exatamente onde e quando cada recurso deve ser alocado. Caso necessário, é possível gerar novos cronogramas com base na mesma probabilidade.

Módulo de output

O módulo de output apresenta o cronograma gerado para o usuário, que pode revisá-lo e aceitá-lo, ou então adicionar novas restrições e reiniciar o processo.

4.5 Avaliação dos resultados

O Federal Air Marshals Service conduziu uma avaliação interna do IRIS e, segundo os autores, os requisitos foram cumpridos, de forma que os *marshals* passaram a ser alocados com base no algoritmo de forma piloto. Em paralelo, o ARMOR foi utilizado por dois anos antes da publicação do artigo com os resultados a seguir [9], de forma que foi possível aos autores realizar análises mais extensivas de sua performance.

A figura 4.5 indica o número de violações encontradas no ano anterior ao uso ARMOR e durante 2008, quando o programa estava em uso. Embora o número encontrado tenha sido substancialmente maior após o emprego do programa, os autores destacam que haviam diversas variáveis que não foram controladas; por exemplo, o número de *checkpoints* não era consistente durante o período.

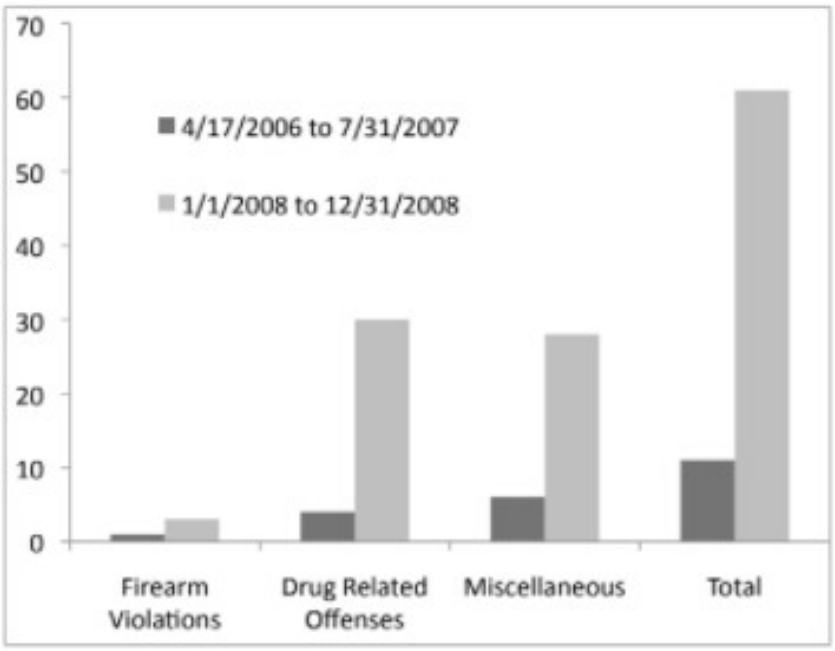


Figura 4.5: *Número de violações detectadas nos checkpoints do aeroporto de Los Angeles antes e depois do DOBSS. Retirado de [9]*

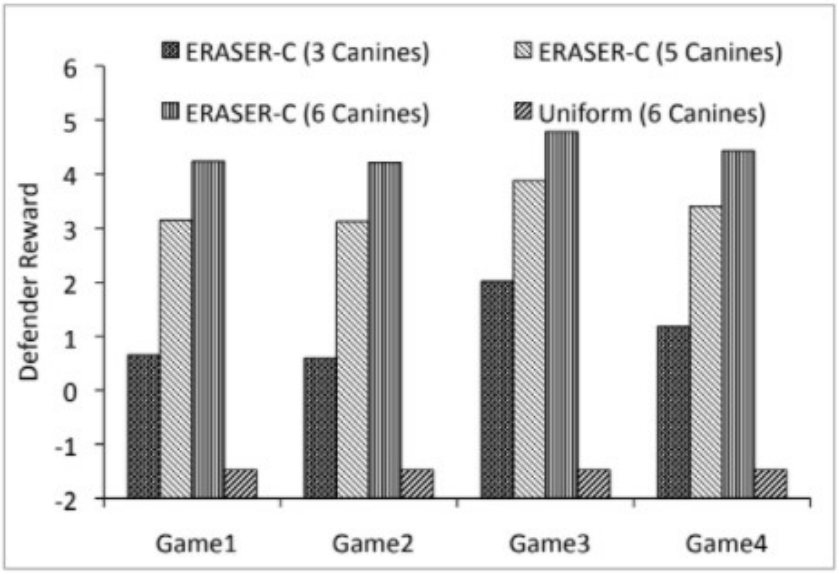


Figura 4.6: *Comparação da recompensa entre as soluções do* ERASER-C *e do modelo uniforme para o problema da patrulha canina, retirada de [9]*

Uma análise comparativa entre os cronogramas desenvolvidos por um modelo aleatório uniforme e pelos métodos do projeto ARMOR para alocar patrulhas caninas nos terminais do LAX realizada pelos autores [9] pode ser vista na figura 4.6. Essa análise consistiu na diferença entre os pagamentos obtidos pela defesa considerando os requisitos do Equilíbrio forte de Stackelberg, em que o atacante sempre desempata a favor da defesa.

Na figura, percebe-se a diferença da recompensa esperada caso se faça uso do ERASER-C com diferentes parâmetros ou de um modelo aleatório uniforme. Nessa análise, os autores utilizaram os modelos reais implementados no ARMOR e, para o modelo aleatório uniforme, patrulhas caninas foram selecionadas aleatoriamente para cada terminal com igual probabi-

lidade.

Assim, percebe-se que o ERASER-C performa melhor mesmo com apenas 3 unidades caninas sendo alocadas entre os terminais quando comparada com 6 unidades do modelo uniforme.

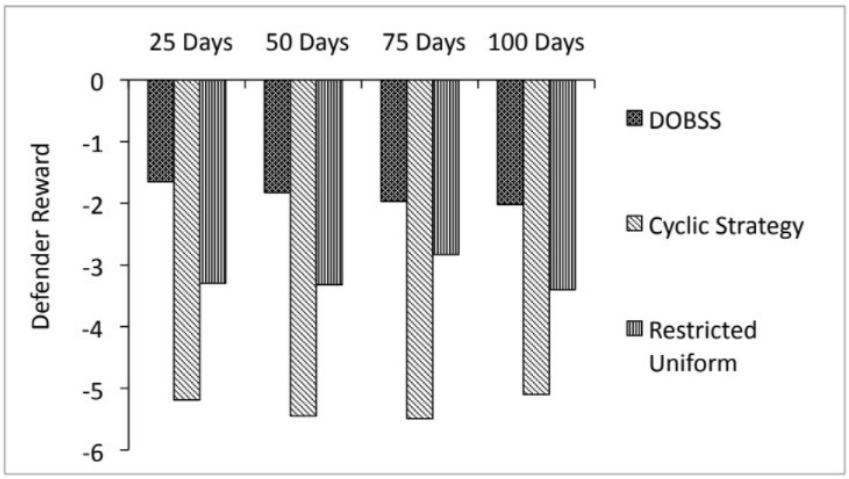


Figura 4.7: *Comparação entre a estratégia* ARMOR *e as políticas representativas de métodos previamente aplicados. Retirado de [9]*

Já na figura 4.7, podemos perceber uma comparação das recompensas obtidas pela defesa na geração de cronogramas utilizando o modelo DOBSS, um modelo cíclico em que *checkpoints* eram alocados em uma ordem cíclica em todas as vias de entrada do LAX, e uma estratégia restritamente uniforme, em que a alocação de *checkpoints* era gerada através de um modelo aleatório uniforme com a restrição de que um *checkpoint* não poderia ser repetido em um intervalo de 2 dias consecutivos.

Para a obtenção desses dados, os autores realizaram uma simulação de um *checkpoint* dentre as cinco vias de entrada do aeroporto com recompensas aleatórias variando entre um e dez. Coletou-se dados sob a premissa estabelecida de que o adversário pôde observar a movimentação da defesa por períodos de 25, 50, 75 e 100 dias, gerando no gráfico a média desses valores sob todos os 100 dias. Podemos observar na figura 4.7 então que a estratégia usada no ERASER-C possui as maiores médias de recompensa obtida pelo fato de que os outros dois modelos são mais previsíveis, em que técnicas simples de reconhecimento de padrões são suficientes para o atacante tomar conhecimento das estratégias usadas e explorá-las. É válido mencionar que os valores das médias das recompensas são negativos por se tratar de uma simulação observando apenas um *checkpoint* com um cronogramada estabelecido previamente.

Os autores também avaliaram a performance dos algoritmos no caso bayesiano, em que a defesa tinha que proteger múltiplos alvos de múltiplos tipos de atacantes.

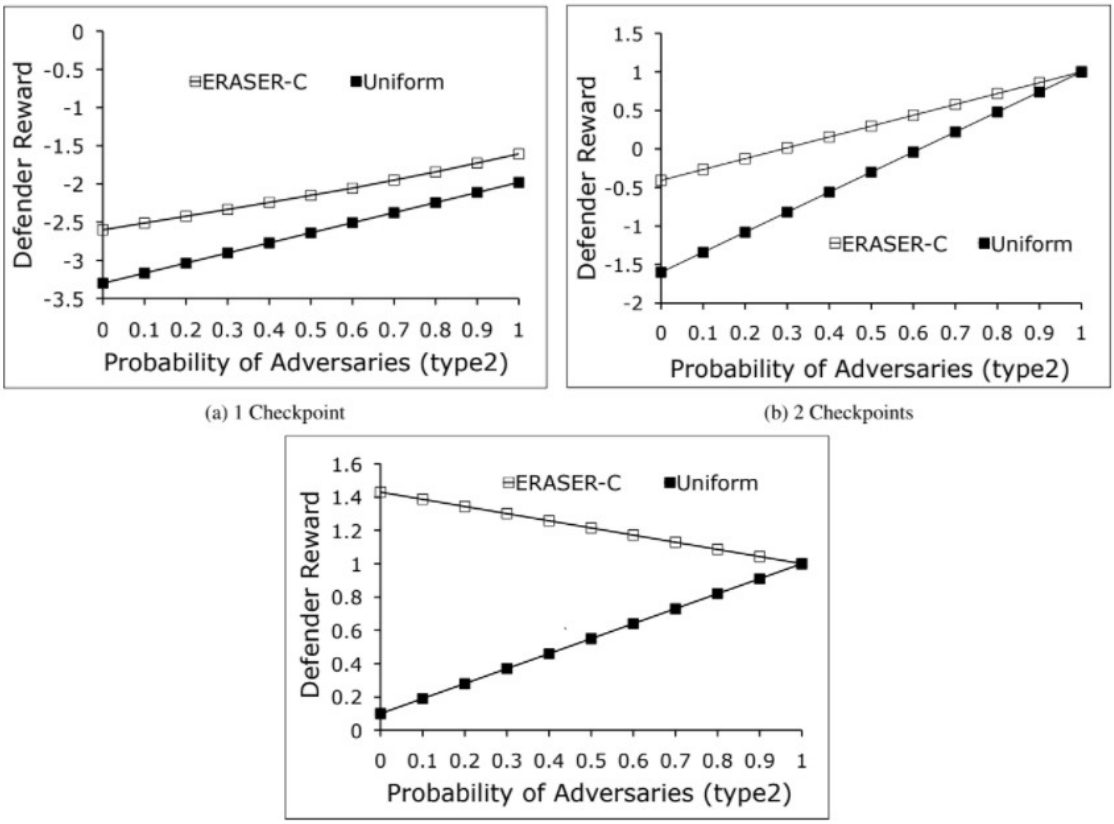


Figura 4.8: *Peformance das estratégias no caso Bayesiano. Retirado de [9]*

A figura 4.8 apresenta os resultados da simulação realizada pelos autores para para a proteção de dez alvos de dois tipos de atacantes por um dois ou três *checkpoints* através da estratégia ERASER-C e de uma estratégia uniforme. Vemos a probabilidade p, no eixo x, do atacante do tipo 2 atacar (com o atacante tipo 1 tendo probabilidade $1 - p$). A figura 4.8 (a) apresenta os resultados para apenas 1 *checkpoint*, enquanto que as figuras 4.8 (b) e (c) mostram os resultados com dois e três *checkpoints*, respectivamente.

É possível perceber que, para qualquer quantidade de *checkpoints* e para qualquer probabilidade do tipo de atacante, a estratégia ERASER-C sempre se mostra superior à estratégia uniforme. Vemos uma exceção quando $p = 1$, ou seja, quando o atacante do tipo 2 está presente para dois e três *checkpoints*, pois, nesse caso, a estratégia ERASER-C escolhe a estratégia *não atacar*. Podemos notar também que os valores das recompensas são sempre positivas na figura 4.8(c), mostrando que, com três *checkpoints* , as chances de capturarem o atacante do tipo 1 aumentam significativamente. Em decorrência disso, é possível notar que a recompensa para o ERASER-C diminui à medida que a probabilidade de que o atacante do tipo 1 apareça também diminui.

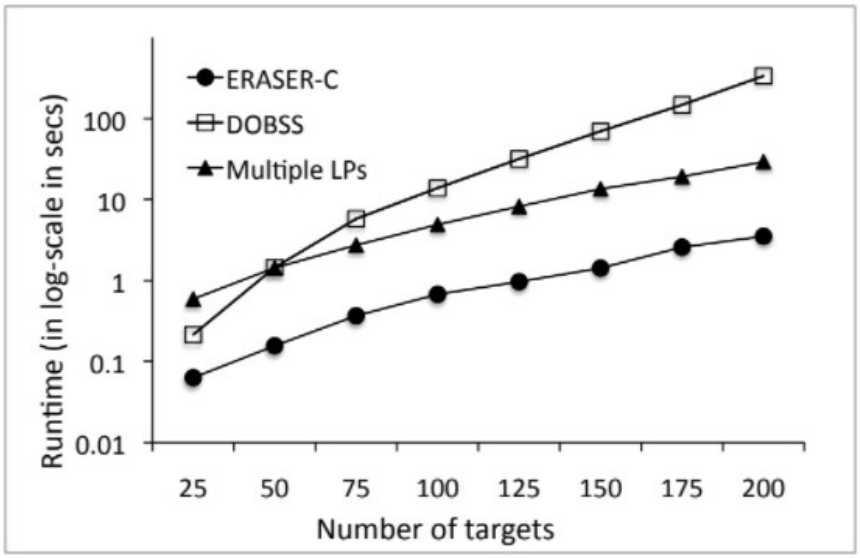


Figura 4.9: *Relação entre o tempo de execução e o quantidade de alvos. Retirado de [9]*

Por fim, os autores compararam o tempo de execução entre os modelos variando a quantidade de alvos para defender, o número de recursos disponíveis e a quantidade de adversários presentes, obtendo assim o tempo médio de execução após 30 simulações com matrizes de jogos geradas aleatoriamente. Na figura 4.9, o número de recursos e a quantidade de adversários foram fixados ambos em 1, enquanto variou-se o número de alvos entre 25 e 200. Podemos perceber que o ERASER-C obteve a melhor performance quando comparado com o DOBSS e com um programa linear múltiplo.

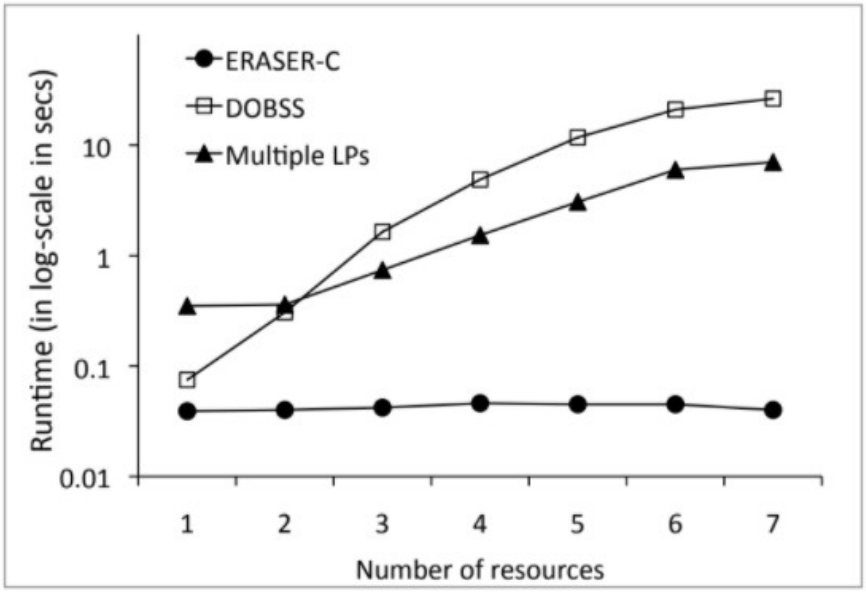


Figura 4.10: *Relação entre o tempo de execução e a quantidade de recursos. Retirado de [9]*

Essa comparação manteve-se na figura 4.10, onde os autores fixaram o número de recursos em 15 e a quantidade de adversários em 1, variando a quantidade de recursos entre 1 e 7. É válido notar que nela e na figura 4.9, o ERASER-C possui um desempenho melhor com uma certa margem de diferença ao comparar os outros 2 modelos. Em compensação, aqui o tempo de execução manteve-se constante mesmo com o tempo estando em uma escala logarítimica,

enquanto que os outros 2 algoritmos apresentaram um incremento a medida que o número de recursos aumentou.

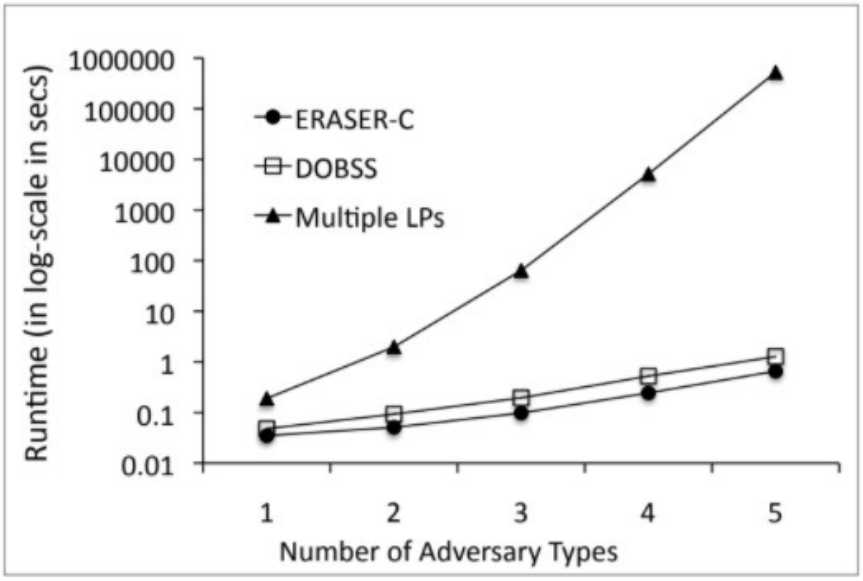


Figura 4.11: *Relação entre o tempo de execução e a quantidade de adversários. Retirado de [9]*

O ERASER-C continuou apresentando o menor tempo médio de execução na figura 4.11, em que os autores variaram a quantidade de adversários entre 1 e 5, fixando a quantidade de recursos em 1 e número de alvos em 10. Aqui o DOBSS já teve um resultado próximo do obtido pelo ERASER-C, enquanto que o método de programação linear múltipla diverge bastante no tempo de execução das outras por usar a transformação de Harsanyi, aumentando bastante seu tempo de execução à medida em que se aumenta a quantidade de adversários.

Capítulo 5

Jogos de Segurança Ambiental

Após o sucesso da implementação de jogos de segurança no domínio de infraestrutura, pesquisadores ampliaram o escopo desses jogos para outros domínios, sendo um deles a supressão de crimes ambientais, como a extração de madeira ilegal de áreas florestais, a caça furtiva em áreas de proteção ambiental e pesca ilegal em áreas de proteção marinhas.

Considere o seguinte exemplo retirado de Fang et al. [5]:

> Um patrulhador ambiental está protegendo uma área de proteção que é povoada por rinocerontes. A área é subdivida em duas subáreas, N_1 e N_2 de igual importância. A patrulhadora escolhe uma subárea para monitorar todos os dias, e ela consegue evitar todo o tipo de armadilha de caçadores na área em que está monitorando. Até então, ela esteve utilizando uma estratégia uniformemente variada. Neste mês, ela pode escolher entre seguir com a estratégia uniformemente variada, capturando 50% das armadilhas. No entanto, os caçadores mudam sua estratégia a cada duas semanas, com base no que observam das atitudes da patrulhadora nas semanas anteriores. Assim, caso a patrulhadora decidisse proteger a área N_1 todos os dias nas primeiras duas semanas no mês e protegesse sempre a área N_2 nas duas semanas restantes, ela capturaria 75%, e não mais 50% das armadilhas – isso porque nas duas primeiras semanas, os caçadores usariam seu conhecimento do mês anterior e seriam indiferentes entre as duas áreas, e colocariam 50% das armadilhas na área N_1 na primeira semana. Mas, a partir da semana 3, eles passariam a caçar sempre na área N_2, com base no conhecimento de que a patrulhadora estará sempre na área N_1, e ela capturará 100% das armadilhas, chegando a um agregado de 75% das armadilhas no mês.

Figura 5.1: *Armadilha usada na caça ilegal de rinocerontes. Retirado de [5]*

Evidentemente, a dinâmica geral é muito semelhante à das situações enfrentadas pelos jogos de segurança apresentados nos capítulos anteriores. No entanto, há especificidades do

domínio ambiental que implicam em alterações de modelagem. Fang e Nguyen [4] detalham algumas das diferenças chaves:

A principal é a frequência e repetição dos ataques – um caçador ou pescador ilegal, por exemplo, coloca armadilhas frequentemente, o que se desdobra em diversas consequências. A primeira dessas consequências é que o jogo deixa de ser um jogo de tentativa única. A segunda é que há um volume maior de dados de ataque que podem ser utilizados pelo defensor na definição de sua estratégia. Já a terceira é que com a frequência, torna-se impossível o atacante realizar vigilância e planejamento de logo prazo antes de cada ataque. A quarta, correlacionada, é que como há um custo relativamente baixo de uma tentativa falha, o atacante empenha menos esforço no planejamento dos ataques, conceito chamado de racionalidade limitada.

Há, ainda, outras diferenças que não decorrem da frequência. As áreas de proteção ambiental, em geral, envolvem grandes áreas de cobertura, muitas vezes com restrições de locomoção devido a características da geografia física do local. Por fim, podem haver restrições na formação de times de patrulha, que muitas vezes envolvem perfis diferentes de profissionais, como policiais, soldados, cientistas e ativistas.

Gholami et al. [6] acrescentam ainda mais uma complexidade à lista: adversários individuais, muitas vezes, se beneficiam com a formação de conluio contra o defensor, coordenando ações para obter melhores informações, reduzir custo de transporte ou atingir novas áreas.

Nas seções seguintes, exploraremos as bases para a modelagem de jogos de segurança ambiental (JSAs).

5.1 Definições dos jogos de segurança ambiental

O trabalho de Fang et al. [5] define jogos de segurança ambiental da seguinte maneira:

Definição 5.1 *Um JSA é um jogo de* $(T < \infty)$ *rodadas repetidas entre uma defensora e* L *atacantes, em que:*

- *a defensora tem* K *guardas para proteger* $N \geq K$ *alvos;*
- *cada rodada tem múltiplos episódios, e em cada episódio, cada guarda pode proteger um alvo e cada atacante pode atacar um alvo;*
- *No início da rodada* t*, a defensora escolhe uma estratégia mista que é uma distribuição de probabilidade sobre todas as estratégias puras, i.e.* $\binom{N}{K}$ *alocações dos guardas nos alvos. Em cada episódio, os guardas protegem alvos de acordo com uma seleção aleatória a partir da estratégia mista;*
- *Cada alvo* $i \in [N]$ *tem valores de pagamento* P_i^a*,* R_i^a*,* P_i^d*,* R_i^d *('P' para pena, 'R' para recompensa, 'a' para atacante e 'd' para defesa). Se um atacante ataca um alvo* i *que está protegido por um guarda, o atacante recebe* P_i^a *e a defesa recebe* R_i^d*, e se o alvo não está protegido, o atacante recebe* R_i^a *e a defesa recebe* P_i^d*.* $R_i^d > P_i^d$ *e* $R_i^a < P_i^a$*;*
- *O pagamento da defesa na rodada* t *é o pagamento esperado calculado para todos os atacantes.*

Cada rodada do jogo corresponde a um período de tempo – por exemplo, uma semana ou um mês – depois do qual a força de defesa – por exemplo, uma agência ambiental ou administradora do parque – se comunica com os guardas locais para atribuir a eles uma nova

estratégia. Cada rodada é dividida entre episódios múltiplos que correspondem a intervalos de tempo menores – por exemplo, dias – em que os jogadores realizam suas ações.

Tipicamente, na literatura de JSAs, a área protegida é dividida em subáreas, de forma que cada uma delas é tratada como um alvo. Diferentes alvos podem ter importâncias distintas para a defensora e para o atacante, devido à riqueza de recursos ou acessibilidade, por exemplo. Assim, cada alvo i recebe um pagamento diferente. Uma estratégia mista da defesa pode ser representada de maneira compacta por meio de um vetor de cobertura $c = \langle c_i \rangle$ em que $0 \leq c_i \leq 1$ é a probabilidade de que o alvo i estará protegido por algum guarda, satisfazendo $\sum_{i=1}^{N} c_i \leq K$, de maneira análoga ao apresentado no capítulo a respeito do ERASER-C. Se um atacante ataca o alvo i, o pagamento esperado da defesa é $U_i^d(c) = c_i R_i^d + (1 - c_i) P_i^d$ para a estratégia c. A estratégia mista da defesa na rodada t será denotada c^t.

T, N, K	número de rodadas, alvos e guardas, respectivamente
L	número de atacantes
c^t	estratégia da defesa na rodada t
$U_i^d(c)$	Pagamento esperado da defesa do alvo i
R_i^d	Recompensa para a defesa do alvo i
P_i^d	Penalidade para a defesa do alvo i

Tabela 5.1: *Quadro resumo da notação utilizada em JSAs*

5.2 O caso PAWS

O PAWS (Protection Assistant for Wildlife Security) foi uma das primeiras aplicações práticas dos jogos de segurança ao domínio da proteção ambiental [22], com o objetivo de auxiliar agências de conservação a ampliar a eficiência das patrulhas de forma que os caçadores ilegais, por medo de serem pegos, sejam desencorajados de caçar furtivamente no Parque Nacional Queen Elizabeth (QENP), em Uganda. Nesse parque, a caça furtiva é realizada principalmente por meio de armadilhas de arame farpado, que são armadas e deixadas sem supervisão, com o caçador retornando depois de um período de tempo para verificar se um animal foi capturado. Os caçadores monitoram as atividades de patrulhamento, recebendo auxílio das aldeias próximas ao parque.

Os patrulhadores do QENP são sobrecarregados – um estudo de 2007 [8] estimou que há um agente para cada 167 quilômetros quadrados.

Definição do problema

Os autores definem como objetivo das patrulhas ambientais prevenir a caça furtiva. Durante uma patrulha, os agentes buscam sinais de atividade ilegal, confiscam as armadilhas encontradas e realizam a apreensão de pessoas que estejam no parque ilegalmente. Além disso, durante as patrulhas, os agentes coletam dados sobre quaisquer atividades ilegais observadas ou suspeitas. Na maioria dos casos, quando um agente encontra uma armadilha, ele não encontra o caçador que a configurou, e isso é chamado de armadilha anônima. Por outro lado, quando um agente encontra e apreende um caçador, muitas vezes ele é capaz de fazer com que o caçador admita onde estão suas armadilhas, que passam a ser armadilhas identificadas. Quando os agentes retornam para a sua base, os dados coletados são transferidos para meios eletrônicos e analisados, de forma que podem ser utilizados futuramente para seguir melhorando as estratégias de patrulhamento.

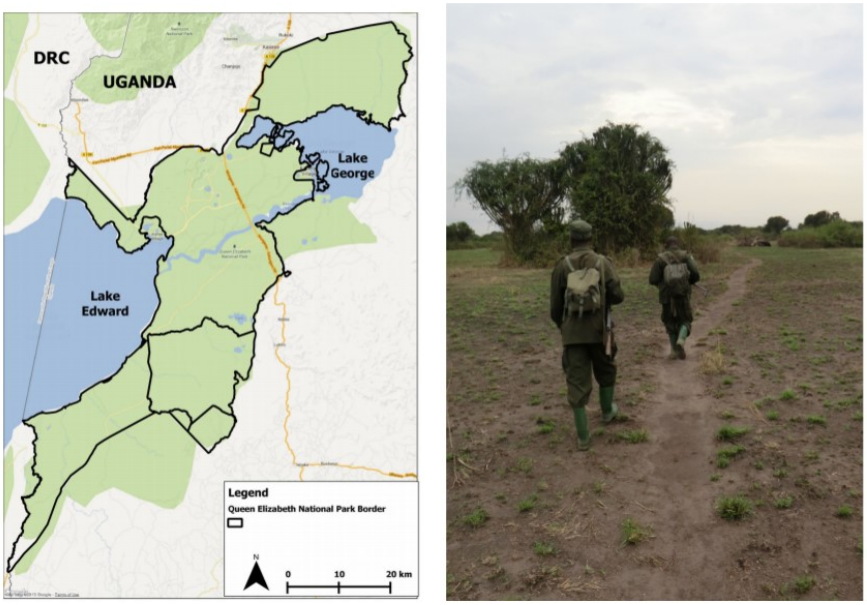


Figura 5.2: *O Parque Nacional Queen Elizabeth, em Uganda, e agentes ambientais em ação no parque. Retirado de [22]*

Do outro lado, o objetivo dos caçadores é realizar com sucesso a caça furtiva. Considera-se como um ataque qualquer armadilha que seja deixada no parque, independente de se um animal foi capturado ou não.

Na modelagem proposta pelo PAWS, assume-se que há apenas um grupo de líderes, ou seja, um grupo de agentes ambientais atuando coordenadamente, com diversos seguidores que atuam independentemente, sem colaboração ou troca de informações.

A área do parque foi discretizada em células de 1 km^2. As diferenças de terreno são importantes, já que em terrenos de vegetação menos densa ou menor variação de relevo, o trajeto pode ser percorrido com mais facilidade e/ou maior velocidade. Além disso, áreas com maior densidade de animais, como regiões próximas de rios ou lagos, apresentam maior propensão para a caça furtiva, e áreas mais distantes da fronteira internacional são menos propensas – uma vez que, como os caçadores precisam carregar os animais capturados para fora do parque, quanto maior a distância, maior a chance de serem apreendidos.

O pagamento é modelado da seguinte forma: se os caçadores atacam uma área coberta, eles recebem um pagamento negativo fixo, que representa uma multa. Há ainda uma penalidade adicional variável, que aumenta conforme a distância do seu ponto de partida. Já os patrulhadores recebem um pagamento positivo fixo, baseado na multa paga pelos caçadores, sem o componente variável baseado na distância – uma atividade de patrulha é considerada um sucesso independentemente de onde são encontrados os caçadores.

Os autores utilizam a representação compacta do ERASER-C, descrita no capítulo anterior, sendo $x =< x_i >$ a estratégia dos patrulhadores, com x_i representando a probabilidade do alvo i estar coberto. A estratégia dos caçadores é dada por $q =< q_i(\omega, x) >$, que articula o comportamento do caçador, ω, com a estratégia dos patrulhadores para resultar na probabilidade de ataque do alvo i.

Conforme exposto anteriormente, uma particularidade dos jogos de segurança ambiental é que eles deixam de ser jogos de tentativa/rodada única. Os pesquisadores modelam a repetição de ataques dos caçadores da seguinte forma: a cada rodada de interação entre os patrulhadores e os caçadores, os patrulhadores executam a mesma estratégia mista. Os caçadores realizam o monitoramento para aprender a estratégia e então respondem. Se os patrulhadores alteram sua estratégia, uma nova rodada começa. Os caçadores tomam suas

decisões baseadas em seu conhecimento da estratégia dos patrulhadores na rodada atual, e assume-se que seu monitoramento fornece conhecimento completo da estratégia de defesa.

Comportamento dos caçadores

Um grande diferencial do PAWS é propor um modelo estocástico para o comportamento dos caçadores, que captura a heterogeneidade no processo de tomada de decisão entre uma população de caçadores.

O modelo baseia-se no modelo SUQR [12], que substitui a função de pagamento esperado por uma função de pagamento subjetiva:

$$SU_i(\omega) = \omega_1 x_i + \omega_2 U^u_{p,i} + \omega_3 U^c_{p,i}$$

onde o parâmetro $\omega = [\omega_1, \omega_2, \omega_3]$ mede o peso de cada fator no processo de tomda de decisão do adversário. Em trabalhos anteriores dos autores, ω era aprendido com base em dados coletados por meio da plataforma da *Amazon Mechanical Turk* e com o algoritmo de *Maximum Likelihood Estimate* (MLE), enquanto no PAWS ele é calculado por uma distribuição de probabilidade.

A ideia central de utilizar uma função de pagamento subjetiva é que indivíduos fazem suas próprias avaliações de cada alternativa durante seus processos de decisão [12].

Essencialmente, o modelo SUQR com um valor específico de ω representa um tipo de caçador. Utilizando uma distribuição de probabilidade contínua em ω, o modelo é transformado em um jogo de Stackelberg Bayesiano com infinitos tipos de adversários. A função de densidade da probabilidade de ω é denotada $f(\omega)$.

O modelo SUQR é uma modificação do modelo *Quantal Response* (QR) [12]. O modelo QR prediz a distribuição estocástica da resposta dos adversários. O parâmetro chave desse modelo é denotado λ, que representa o nível de racionalidade da resposta do adversário: quanto maior λ, mais a resposta prevista pelo modelo converge para a resposta ótima do adversário.

No modelo QR, a probabilidade de que o adversário ataque o alvo t, denotada q_t é dada por

$$q_t = \frac{e^{\lambda U^a_t}}{\sum_{t'} e^{\lambda U^a_{t'}}}$$

em que U^a_t é a recompensa recebida pelo atacante ao escolher o alvo t. Já no SUQR, substituimos U^a_t pela função de pagamento subjetiva SU_i:

$$q_t = \frac{e^{\lambda(\omega_1 x_i + \omega_2 U^u_{p,i} + \omega_3 U^c_{p,i})}}{\sum_{t'} e^{\lambda(\omega_1 x_i + \omega_2 U^u_{p,i} + \omega_3 U^c_{p,i})}}$$

Caso tivéssemos o modelo de comportamento de toda a população de adversários, a estratégia ótima de patrulhamento $x*$ seria a que maximiza a utilidade esperada dos patrulhadores:

$$x* = \arg\max_x \oint U_r(x|\omega) f(\omega) d\omega$$

em que $U_r(x|\omega)$ é a utilidade esperada dos patrulhadores ao executar a estratégia x contra um caçador que tem um parâmetro do modelo ω, e é computado da seguinte maneira:

$\mathbb{T}$	conjunto de alvos; $i \in \mathbb{T}$ denota o alvo i
x_i	Probabilidade do alvo i estar protegido
$U^c_{r,i}$	Pagamento recebido pela defesa por proteger o alvo i caso seja selecionado pelos atacantes
$U^u_{r,i}$	Pagamento recebido pela defesa por não proteger o alvo i caso seja selecionado pelos atacantes
$U^c_{p,i}$	Pagamento recebido pelo atacante por selecionar o alvo i caso esteja protegido pela defesa
$U^u_{p,i}$	Pagamento recebido pelo atacante por selecionar o alvo i caso não esteja protegido pela defesa
ω	Parâmetro do modelo SUQR
$f(\omega)$	Função de densidade de ω
$U_r(x\|\omega)$	Utilidade esperada dos patrulhadores ao executar a estratégia x contra um caçador que tem um parâmetro do modelo ω
$U_r(x)$	Utilidade esperada dos patrulhadores ao executar a estratégia x contra toda a população de caçadores
$q_i(\omega\|\mathbb{G})$	Probabilidade de que o caçador com parâmetro ω selecione o alvo i no jogo $\mathbb{G}$

Tabela 5.2: *Quadro resumo da notação utilizada no PAWS*

$$U_r(x|\omega) = \sum_i U_{r,i}(x|\omega) q_i(\omega|\mathbb{G})$$

em que $U_{r,i}(x|\omega)$ é a utilidade esperada dos patrulhadores se o alvo i for selecionado pelos caçadores. $U_r(x|\omega)$ é uma função não-linear fracional, uma vez que $q_i(\omega|\mathbb{G})$ segue a predição do modelo SUQR.

O cerne do problema é que, na verdade, o modelo de comportamento da população de caçadores é desconhecida para a patrulha. Assim, o principal desafio em obter uma estratégia de patrulha ótima é aprender o modelo de comportamento dos caçadores e, mais especificamente, a distribuição do parâmetro ω do modelo do SUQR.

Aprendendo o modelo de comportamento

No começo do jogo, os patrulheiros sabem apenas que a distribuição dos parâmetros do modelo de comportamento da população de caçadores se comportam como uma distribuição normal. O objetivo é então descobrir a média μ e a matriz de covariância Σ do parâmetro ω do SUQR à medida que os dados ficam disponíveis.

Como visto anteriormente, situações em que os caçadores são capturados são escassas e, por isso, leva-se muito mais tempo para coletar dados suficientes para se construir um modelo de distribuição razoável. Entretanto, há muitos dados de caça ilegal anônima e ambos os tipos de dados serão utilizados em conjunto para uma melhora na convergência do aprendizado.

A cada rodada t do jogo, são observados $N_a^{(t)}$ crimes anônimos e capturados $N_c^{(t)}$ caçadores ilegais. Iremos denotar por $A^{(t)} = \{a_j^{(t)} | j = 1, \ldots, N_a^t\}$ o conjunto de alvos escolhidos pelos caçadores que cometeram crimes anônimos, e por $\Omega^{(t)} = \{\omega_k^{(t)} | j = 1, \ldots, N_c^t\}$ os valores dos parâmetros do SUQR associados aos caçadores capturados. Assume-se que um caçador capturado irá confessar todo o seu histórico de crimes cometidos até o momento de sua captura, de forma que, para o caçador k, denotamos por $C_k^{(t)} = \{c_{k,l}^{(t)}\}$ o conjunto de crimes cometidos por ele, em que o índice l representa o l-ésimo crime cometido por ele, e por

$c_{k,l}^{(t)} = (\alpha_l^{(t)}, x_l^{(t)})$ o crime, sendo que $\alpha_{k,l}^{(t)}$ denota seu alvo e $x_{k,l}^{(t)}$ denota a alocação dos recursos de patrulhamento naquele momento. Para uma questão de simplificação, $\alpha_{k,l}^{(t)}$ será chamado de $\alpha_{k,l}$ e $x_{k,l}^{(t)}$ de x_l.

Para cada caçador capturado, seu parâmetro ω será estimado por meio de *Maximum Likelihood Estimate* da seguinte maneira:

$$\omega_k^{(t)} = \arg\max_w \log L(\omega|C_k^{(t)}) = \arg\max_w \sum_l \log(q_{a_l}(\omega|x_l))$$

em que $q_{a_l}(\omega|x_l)$ é a probabilidade prevista de que o k-ésimo caçador capturado escolha o alvo α_l ao cometer um crime depois de observar a estratégia x_l de alocação dos patrulhadores. Após a rodada t, há $\sum_{\tau=1}^{t} N_c^{(\tau)}$ caçadores capturados e, portanto, conjuntos de dados. Ao aplicar *Maximum Likelihood Estimate(MLE)*, é possível aprender o parâmetro ω com essas amostras. Sabendo que ω segue uma distribuição normal de 3 dimensões, a média e a matriz de covariância aprendida via *MLE* são calculadas como exposto abaixo:

$$\mu^{(t)} = \frac{1}{\sum_{\tau=1}^{t} N_c^{(\tau)}} \sum_{\tau=1}^{t} \sum_{\omega \in \Omega^{(\tau)}} \omega$$

$$\sum^{(\tau)} = \frac{1}{\sum_{\tau=1}^{t} N_c^{i}} \sum_{\tau=1}^{t} \sum_{\omega \in \Omega^{i}} (\omega - \mu^{(t)})(\omega - \mu^{(t)})^T$$

No caso dos crimes anônimos, não é possível estimar parâmetros individualmente como para os caçadores capturados. No entanto, podemos tratar cada um dos dados sobre crimes anônimos como tendo sido cometidos por caçadores diferentes, e depois prosseguir de maneira similar com o *MLE*.

$$\omega_j^{(t)} = \arg\max_w \log L(\omega|a_j^{(t)}, x^{(t)})$$

Combinando dados anônimos e dados identificados

Enquanto dados anônimos provocam ruídos no modelo de comportamento de um caçador, eles promovem uma acurácia significativa na distribuição da população de caçadores ilegais devido a sua abundância. *PAWS-Learn* é o nome do algoritmo que melhora a estimativa dos parâmetros do modelo ao combinar dados conhecidos com dados anônimos.

Algorithm 5 PAWS-Learn

Input: t, $C^{(\tau)}$, $A^{(\tau)}$, $x^{(\tau)}$, $\forall \tau \in 1...t-1$;
$(\mu^{(\tau)}, \Sigma) \leftarrow$ Learn($\{C^{(\tau)}, \tau{=}1, ..., t-1\}$);
$(\{\omega_n\}) \leftarrow$ Sample($\mu^{(\tau)}$, $\sum_{(\tau)}$, N_s);
$\pi_n^o \leftarrow \frac{f(\omega_n|\mu^{(t)},\sum^{(t)}}{\sum_{n'} f(\omega_{n'}|\mu^{(t)},\sum^{(t)})}, \forall n$;
$\langle \pi_n \rangle \leftarrow$ Refine($\{C^{(\tau)}\}$, $\tau = 1, \ldots, t-1\}$), $\langle \pi_n^o \rangle$);
Return ($\{\omega_n\}$,$\langle \pi_n \rangle$);

Na rodada t, *PAWS-Learn* primeiro usa os dados já conhecidos para calcular a média e a variância na função *Learn*. Em seguida, ele calcula a acurácia dessa estimativa utilizando o método do erro quadrático médio (*MSE*) na estimativa obtida via distribuição de crime observada pelos dados anônimos.

$$MSE^{(t)}(\mu^{(t)}, \sum^{(t)}) = \sum_{i \in \tau} (\bar{q}_i(x^{(t)}|\mu^{(t)}, \sum^{(t)}) - y_i^{(t)})^2$$

onde $y_i^{(t)}$ representa a proporção de crimes no alvo i registrados pelos dados anônimos. $\bar{q}_i(x^{(t)}|\mu^{(t)}, \sum^{(t)})$ representa a probabilidade de que o alvo i será escolhido pela população de caçadores.

Adaptando a estratégia de patrulha

O *PAWS-Adapt* foi criado posteriormente como uma melhoria do *PAWS-Learn*. Na rodada t, ele estima primeiramente o modelo de comportamento utilizando todo o histórico de dados do *PAWS-Learn*. Ele então computa a estratégia de patrulha ótima baseado no atual resultado do aprendizado para ser executado na próxima rodada. Para isso, ele precisa resolver um problema de otimização equivalente a um jogo de Stackelberg bayesiano com infinitos tipos.

Assim, seja $x^{(t)}$ a estratégia de patrulha computada pelo *PAWS-Adapt* na rodada t. Os patrulheiros irão atualizar o modelo de comportamento dos caçadores a cada rodada após mais dados forem sendo coletados. Os patrulheiros irão assim atualizar também sua estratégia a partir do modelo atualizado.

5.3 Resultados obtidos pelos autores

No primeiro conjunto de experimentos realizado por Yang et al. [22], foi gerada uma matriz de pagamento aleatória. Os dados sobre os crimes foram gerados da seguinte forma: dada a distribuição verdadeira ω, primeiro foi gerado um conjunto de parâmetros aleatórios para ω para representar toda a população de caçadores, com N_p representando o número total deles. A cada rodada, um subconjunto de ω foi gerado para representar os caçadores que realizaram um ataque naquela rodada. Dada a estratégia dos patrulheiros, foram simulados os alvos que os caçadores atacaram e esses dados foram registrados como dados anônimos. Enquanto isso, foram escolhidos aleatoriamente caçadores desse subconjunto para representar os caçadores que foram capturados pelos patrulheiros naquela rodada. As escolhas que esses caçadores capturados tiveram na última rodada são registradas como dados conhecidos.

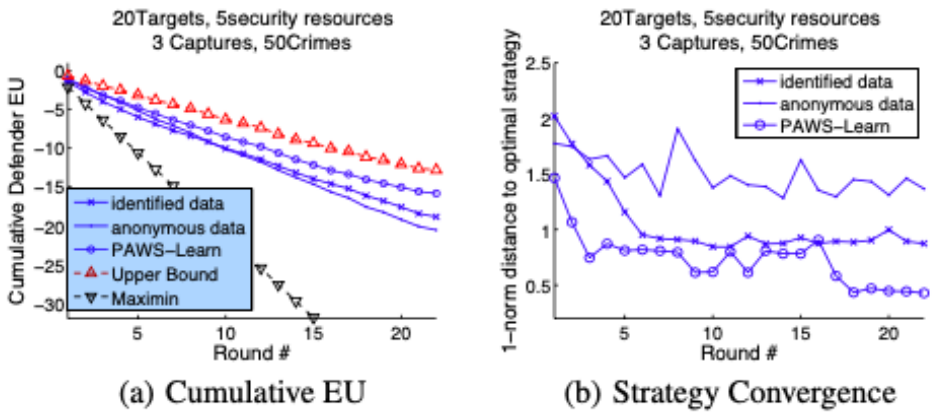


Figura 5.3: *Resultado das simulações ao longo das rodadas. Retirado de [22]*

Na figura 5.3(a), é mostrado o pagamento esperado acumulado pelos patrulheiros ao longo das rodadas. Três métodos foram comparados: o *PAWS-Learn*, o modelo utilizando apenas dados conhecidos e o modelo utilizando apenas dados anônimos. A estratégia utilizando *maximin* foi usada como base de comparação, assim como a estratégia ótima computada caso os patrulheiros soubessem a verdadeira distribuição de ω. Ambas foram incluídas e se encontram presente na figura.

Ainda na figura 5.3(a), é mostrado o resultado médio de 20 instâncias de jogos aleatórios, cuja configuração foi de 20 alvos e 5 recursos de segurança. A cada rodada, 50 dados anônimos foram gerados e 3 caçadores ilegais foram capturados. Como observado na figura, o *PAWS-Learn* obteve a melhor performance dentre os outros 2 modelos. Também podemos observar que os métodos de aprendizado de fato geram resultados ao analisarmos que os 3 modelos estão mais próximos do limite superior do que da estratégia *Maximin*.

Na figura 5.3(b), é mostrada a convergência das estratégias de patrulhamento dos 3 modelos, em que o *PAWS-Learn* atingiu a conversão mais rápido que os outros 2 métodos. Logo, combinar os 2 tipos de dados gera uma melhora no aprendizado do modelo de comportamento de caçadores ilegais.

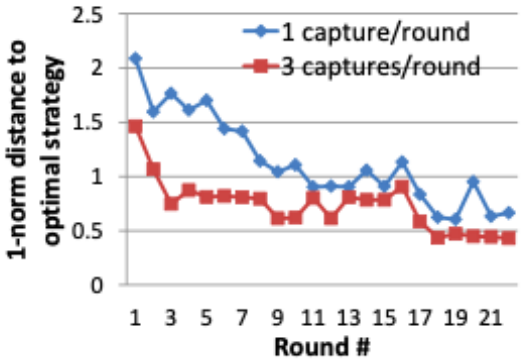
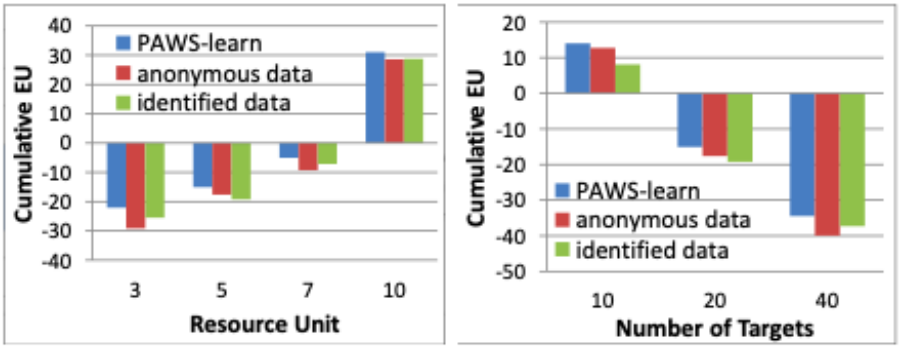


Figura 5.4: *Análise da velocidade de captura. Retirado de [22]*

Yang et al. [22] também realizaram uma comparação entre a velocidade de captura ao longo das rodadas, apresentada na figura 5.4, a fim de analisar o impacto na performance do *PAWS-Learn*. Foi analisada a convergência do modelo ao se capturar 1 ou 3 caçadores ilegais por rodada, resultando em uma convergência mais rápida no caso da captura de 3 caçadores ilegais.



(a) 20 Targets, 3 Captures, 50 Crimes

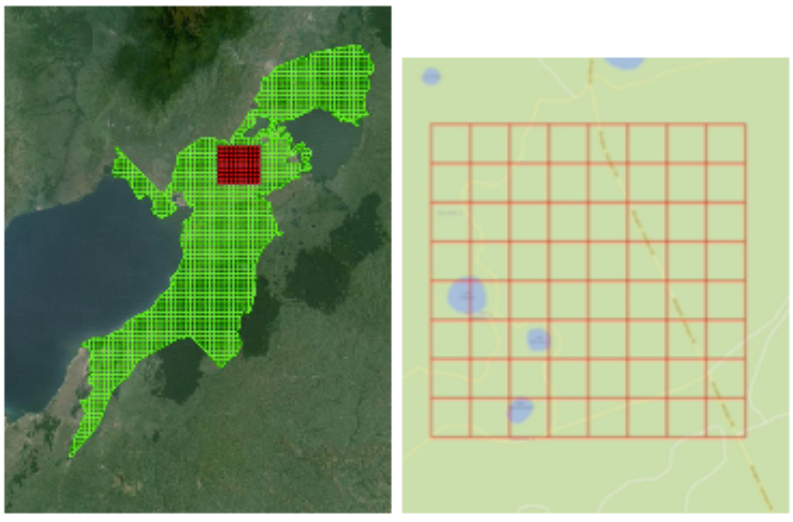
(b) 5 security resources, 3 Captures, 50 Crimes

Figura 5.5: *Análise da recompensa esperada na rodada 20. Retirado de [22]*

Também foi realizada pelos autores uma análise da recompensa esperada variando o número de alvos e de recursos ao final de 20 rodadas, com 50 crimes e capturando 3 caçadores ilegais por rodada. Na figura 5.5(a), varia-se o número de recursos enquanto é fixado o número de alvos em 20. Percebe-se que a recompensa esperada aumenta à medida que aumentamos o número de recursos.

Em contrapartida, ao fixar o número de recursos em 5 e variar o número de alvos, percebe-se, na figura 5.5(b) que a recompensa esperada diminui à medida que aumentamos o número de alvos. Nota-se que *PAWS-Learn* obteve um melhor resultado dentre os outros 2 modelos tanto na 5.5(a) quanto na 5.5(b).

Resultados na área de implementação



(a) A zoomed out view of the simulation area.

(b) The 64 sq. km grid overlayed on the simulation area.

Figura 5.6: *Área do QENP de interesse para a simulação. Retirado de [22]*

Os resultados dos experimentos do *PAWS* são apresentados abaixo, na figura 5.7. Foi selecionado um terreno plano do QENP *(Queen Elizabeth National Park)*, rota internacional de $64km^2$ que conecta a República Democrática do Congo com pequenas rodovias e com dois lagos, mostrado na figura 5.6(a). Na figura figura 5.6(b), destaca-se a região da simulação, com destaque para os lagos, que correspondem a áreas de alta densidade de fauna.

O cálculo da matriz de recompensa dos caçadores ilegais e dos patrulheiros levou em conta primariamente a densidade de fauna das células da 5.6(b), além da distância das rodovias que se encontram mais próximas da célula de interesse. Para um patrulheiro, a sua recompensa permanece uniforme dado que seu interesse é apenas em capturar caçadores ilegais, independente da célula em que se encontram, enquanto que sua penalidade é baseada na densidade animal da célula atacada.

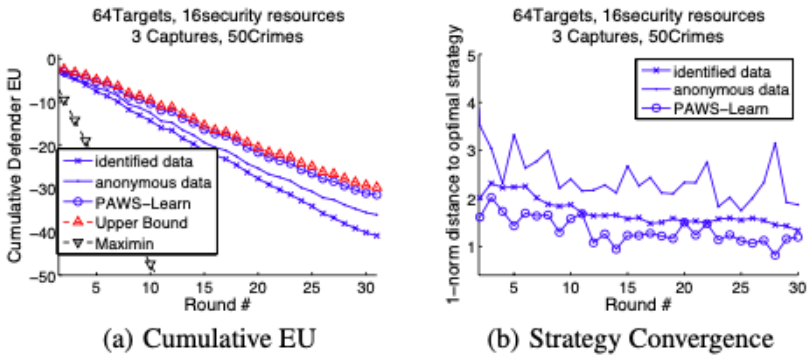


Figura 5.7: *Resultado da simulação no QENP. Retirado de [22]*

Simulações foram realizadas pelos autores sob as mesmas configurações apresentadas na seção 5.3 e seus resultados foram apresentados na figura 5.7. O número de recursos foi fixado em 16, indicando que um patrulheiro cobre 16 células no mapa da figura 5.6(b). A recompensa esperada acumulada é apresentada em 5.7(a) e podemos perceber que o *PAWS-Learn* alcança uma performance muito próxima à estratégia ótima. A convergência da estratégia de patrulhamento é apresentada em 5.7(b).

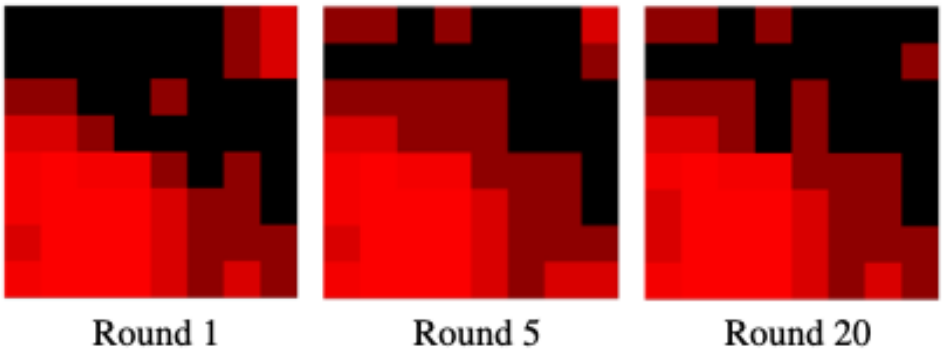


Figura 5.8: *Densidade da cobertura do patrulhamento. Retirado de [22]*

Para facilitar a visualização da estratégia de patrulhamento mudando ao longo das rodadas, vemos na figura 5.8 a densidade de patrulhamento ao longo da matriz 8x8 em 3 rodadas diferentes: na rodada 1, 5 e 20. As áreas em vermelho indicam alta densidade animal, como pode ser percebido no canto inferior esquerdo, onde há a presença de lagos, enquanto as áreas pretas indicam baixa densidade.

Conclusões

Neste trabalho, apresentamos os fundamentos teóricos que nos levam dos Jogos de Stackelberg aos Jogos de Segurança Ambiental, passando pelos Jogos de Segurança. Apresentamos abordagens algorítmicas para a sua solução e analisamos uma implementação de uma dessas soluções para o monitoramento da caça ilegal nas florestas de Uganda.

Durante a elaboração da proposta inicial, acreditávamos que eventualmente poderíamos reproduzir a implementação de algum dos algoritmos estudados, seja com o conjunto de dados utilizado pelos pesquisadores, seja em um novo conjunto de dados, como os dados abertos do Ministério do Meio Ambiente. No entanto, isso não foi possível por dois principais motivos. No caso dos conjuntos de dados utilizados pelos pesquisadores, por se tratarem de algoritmos que são, de fato, utilizados em alguns aeroportos americanos e na alocação dos *marshals*, os dados referentes aos aeroportos, aos voos e às matrizes de pagamento não são disponibilizados, o que evidentemente leva também a um problema de reprodutibilidade dos resultados. No caso do PAWS, o conjunto de dados históricos utilizados para determinar a estratégia de patrulhamento também não são disponibilizados. Quanto ao conjunto de dados brasileiros, nos deparamos com dois problemas: ainda que haja séries temporais de fotografias de satélite pelas quais é possível identificar o desmatamento, as técnicas de processamento de sinais digitais que seriam necessárias para transformá-las no *input* requerido adicionariam uma camada de complexidade muito além do escopo deste trabalho e, ainda que ultrapassada essa barreira, não possuíamos o auxílio de especialistas para determinar a matriz de pagamento.

Apesar desses empecilhos, o estudo do conteúdo teórico mostrou-se muito interessante, combinando aprendizados de diversas disciplinas que cursamos ao longo do bacharelado, como Otimização Linear (o teorema Minimax e programação linear), Teoria dos Jogos (definições de jogo, pagamento, equilíbrio de Nash), Aprendizado de Máquina (a modelagem de comportamento da população e aprendizado do parâmetro ω do SUQR) e Inteligência Artificial (problemas de busca adversarial), além dos fundamentos aprendidos nas disciplinas de Algoritmos e Estruturas de Dados, Álgebra Linear, Probabilidade e Estatística, e Análise de Algoritmos. Como um trabalho de formatura, foi muito satisfatório perceber como conteúdos de áreas aparentemente distintas são capazes de se unir em um todo coeso.

Apesar da complexidade do tema, este texto foi concebido para ser um guia introdutório e didático sobre os jogos de segurança e algumas de suas aplicações, contribuindo para que o leitor se familiarize com o assunto. Esperamos que seja útil para aqueles que pretendem aplicar as modelagens dos jogos de segurança no domínio da proteção de infraestutura e, especialmente, no domínio da segurança ambiental.

Apêndice A

Os Objetivos de Desenvolvimento Sustentável da Agenda 2030

Conforme enumerados na Agenda 2030 [21], p. 19, os dezessete Objetivos de Desenvolvimento Sustentável são:

1. Acabar com a pobreza em todas as suas formas, em todos os lugares;
2. Acabar com a fome, alcançar a segurança alimentar e melhoria da nutrição e promover a agricultura sustentável;
3. Assegurar uma vida saudável e promover o bem-estar para todos, em todas as idades;
4. Assegurar a educação inclusiva e equitativa e de qualidade, e promover oportunidades de aprendizagem ao longo da vida para todos;
5. Alcançar a igualdade de gênero e empoderar todas as mulheres e meninas;
6. Assegurar a disponibilidade e gestão sustentável da água e saneamento para todos;
7. Assegurar o acesso confiável, sustentável, moderno e a preço acessível à energia para todos;
8. Promover o crescimento econômico sustentado, inclusivo e sustentável, emprego pleno e produtivo e trabalho decente para todos;
9. Construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável e fomentar a inovação;
10. Reduzir a desigualdade dentro dos países e entre eles;
11. Tornar as cidades e os assentamentos humanos inclusivos, seguros, resilientes e sustentáveis;
12. Assegurar padrões de produção e de consumo sustentáveis;
13. Tomar medidas urgentes para combater a mudança climática e seus impactos;[1]
14. Conservação e uso sustentável dos oceanos, dos mares e dos recursos marinhos para o desenvolvimento sustentável;

[1]Reconhecendo que a Convenção Quadro das Nações Unidas sobre Mudança do Clima (UNFCCC) é o fórum internacional intergovernamental primário para negociar a resposta global à mudança do clima.

15. Proteger, recuperar e promover o uso sustentável dos ecossistemas terrestres, gerir de forma sustentável as florestas, combater a desertificação, deter e reverter a degradação da terra e deter a perda de biodiversidade;

16. Promover sociedades pacíficas e inclusivas para o desenvolvimento sustentável, proporcionar o acesso à justiça para todos e construir instituições eficazes, responsáveis e inclusivas em todos os níveis;

17. Fortalecer os meios de implementação e revitalizar a parceria global para o desenvolvimento sustentável.

Apêndice B

Objetivo 15: Vida Terrestre

O objetivo 15, vida terrestre, é, na sua forma longa, "proteger, recuperar e promover o uso sustentável dos ecossistemas terrestres, gerir deforma sustentável as florestas, combater a desertificação, deter e reverter a degradaçãoda terra e deter a perda de biodiversidade" [21, p.29]. Ele divide-se nas seguintes metas:

1. Até 2020, assegurar a conservação, recuperação e uso sustentável de ecossistemas terrestres e de água doce interiores e seus serviços, em especial florestas, zonas úmidas, montanhas e terras áridas, em conformidade com as obrigações decorrentes dos acordos internacionais;

2. Até 2020, promover a implementação da gestão sustentável de todos os tipos de florestas, deter o desmatamento, restaurar florestas degradadas e aumentar substancialmente o florestamento e o reflorestamento globalmente;

3. Até 2030, combater a desertificação, restaurar a terra e o solo degradado, incluindo terrenos afetados pela desertificação, secas e inundações, e lutar para alcançar um mundo neutro em termos de degradação do solo;

4. Até 2030, assegurar a conservação dos ecossistemas de montanha, incluindo a sua biodiversidade, para melhorar a sua capacidade de proporcionar benefícios que são essenciais para o desenvolvimento sustentável;

5. Tomar medidas urgentes e significativas para reduzir a degradação de habitat naturais, deter a perda de biodiversidade e, até 2020, proteger e evitar a extinção de espécies ameaçadas;

6. Garantir uma repartição justa e equitativa dos benefícios derivados da utilização dos recursos genéticos e promover o acesso adequado aos recursos genéticos;

7. Tomar medidas urgentes para acabar com a caça ilegal e o tráfico de espécies da flora e fauna protegidas e abordar tanto a demanda quanto a oferta de produtos ilegais da vida selvagem;

8. Até 2020, implementar medidas para evitar a introdução e reduzir significativamente o impacto de espécies exóticas invasoras em ecossistemas terrestres e aquáticos, e controlar ou erradicar as espécies prioritárias;

9. Até 2020, integrar os valores dos ecossistemas e da biodiversidade ao planejamento nacional e local, nos processos de desenvolvimento, nas estratégias de redução da pobreza e nos sistemas de contas.

Bibliografia

[1] *Climate Change and Land: an IPCC special report on climate change, desertification, land degradation, sustainable land management, food security, and greenhouse gas fluxes in terrestrial ecosystems*. In Press. IPCC, Outubro de 2019. Cap. Summary for Policymakers. URL: https://www.ipcc.ch/srccl/chapter/summary-for-policymakers/.

[2] Vincent Conitzer. "On Stackelberg mixed strategies". Em: *Synthese* 193.3 (2016), pp. 689–703. URL: https://arxiv.org/abs/1705.07476.

[3] Vincent Conitzer e Tuomas Sandholm. "Computing the Optimal Strategy to Commit To". Em: *Proceedings of the 7th ACM Conference on Electronic Commerce*. EC '06. Ann Arbor, Michigan, USA: Association for Computing Machinery, 2006, pp. 82–90. ISBN: 1595932364. DOI: 10.1145/1134707.1134717. URL: https://doi.org/10.1145/1134707.1134717.

[4] Fei Fang e Thanh H. Nguyen. "Green Security Games: Apply Game Theory to Addressing Green Security Challenges". Em: *ACM SIGecom Exchanges* 15 (jul. de 2016). Ed. por ACM. On-line; acessado em 21-12-2020, pp. 78–83. URL: http://www.sigecom.org/exchanges/volume_15/1/FANG.pdf.

[5] Fei Fang, Peter Stone e Milind Tambe. "When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing". Em: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 2589–2595. ISBN: 9781577357384.

[6] Shahrzad Gholami et al. "Divide to defend: Collusive security games". Em: *International Conference on Decision and Game Theory for Security*. On-line; acessado em 21-12-2020. Springer. 2016, pp. 272–293. URL: https://projects.iq.harvard.edu/files/teamcore/files/divide_to_defend_collusive_security_games.pdf.

[7] John C. Harsanyi. "Games with Incomplete Information Played by Bayesian Players, I-III. Part I. The Basic Model". Em: *Management Science* 14.3 (1967), pp. 159–182. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2628393.

[8] T. HOLMERN, J. MUYA e E. RØSKAFT. "Local law enforcement and illegal bushmeat hunting outside the Serengeti National Park, Tanzania". Em: *Environmental Conservation* 34.1 (2007), pp. 55–63. DOI: 10.1017/S0376892907003712.

[9] Manish Jain et al. "Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshal Service". Em: *Interfaces* 40.4 (jul. de 2010), pp. 267–290. ISSN: 0092-2102. DOI: 10.1287/inte.1100.0505. URL: https://doi.org/10.1287/inte.1100.0505.

[10] P. Morris. *Introduction to Game Theory*. Universitext. New York: Springer, 2012. ISBN: 9781461243168.

[11] John von Neumann. "Zur Theorie der Gesellschaftsspiele". Em: *Mathematische Annalen* 100 (1928), pp. 295–320.

[12] T. Nguyen et al. "Analyzing the Effectiveness of Adversary Modeling in Security Games". Em: *AAAI*. 2013.

[13] Praveen Paruchuri et al. "Playing Games for Security: An Efficient Exact Algorithm for Solving Bayesian Stackelberg Games". Em: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '08. Estoril, Portugal: International Foundation for Autonomous Agents e Multiagent Systems, 2008, pp. 895–902. ISBN: 9780981738116.

[14] James Pita et al. "Deployed ARMOR Protection: The Application of a Game Theoretic Model for Security at the Los Angeles International Airport". Em: AAMAS '08. Estoril, Portugal: International Foundation for Autonomous Agents e Multiagent Systems, 2008, pp. 125–132.

[15] James Pita et al. "GUARDS: Game Theoretic Security Allocation on a National Scale". Em: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '11. Taipei, Taiwan: International Foundation for Autonomous Agents e Multiagent Systems, 2011, pp. 37–44. ISBN: 0982657153.

[16] Stuart J. Russell e Peter Norvig. *Artificial Intelligence: a modern approach*. 3ª ed. Pearson, 2009.

[17] Eric Shieh et al. "PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States". Em: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '12. Valencia, Spain: International Foundation for Autonomous Agents e Multiagent Systems, 2012, pp. 13–20. ISBN: 0981738117.

[18] Heinrich von Stackelberg. *Marktform und Gleichgewicht*. 1st. Berlin: Verlag von Julius Springer, 1934.

[19] *Summary for policymakers of the global assessment report on biodiversity and ecosystem services of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services*. Working Papers. IPBES Secretariat, Outubro de 2019. URL: https://ipbes.net/global-assessment.

[20] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. 1st. Estados Unidos: Cambridge University Press, 2011. ISBN: 1107096421.

[21] *Transformando Nosso Mundo: A Agenda 2030 para o Desenvolvimento Sustentável*. Working Papers. Organização das Nações Unidas, Outubro de 2015. URL: https://nacoesunidas.org/wp-content/uploads/2015/10/agenda2030-pt-br.pdf.

[22] Rong Yang et al. "Adaptive Resource Allocation for Wildlife Protection against Illegal Poachers". Em: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*. AAMAS '14. Paris, France: International Foundation for Autonomous Agents e Multiagent Systems, 2014, pp. 453–460. ISBN: 9781450327381.

[23] Zhengyu Yin et al. "TRUSTS: Scheduling Randomized Patrols for Fare Inspection in Transit Systems". Em: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI'12. Toronto, Ontario, Canada: AAAI Press, 2012, pp. 2348–2355.