POLITECHNIKA WARSZAWSKA

INFORMATYKA TECHNICZNA I TELEKOMUNIKACJA
DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

Rozprawa doktorska

mgr inż. Adam Żychowski

Zastosowanie algorytmów ewolucyjnych w wielokrokowych Grach Obronnych Stackelberga

Promotor
prof. dr hab. inż. Jacek Mańdziuk

WARSZAWA 2022

Streszczenie

Tytuł: *Zastosowanie algorytmów ewolucyjnych w wielokrokowych Grach Obronnych Stackelberga*

W rozprawie zbadana została możliwość zastosowania algorytmów ewolucyjnych do poszukiwania aproksymacji stanu równowagi w wielokrokowych grach obronnych Stackelberga (ang. *Stackelberg Security Games*). Gry obronne Stackelberga rozgrywane są pomiędzy dwoma niesymetrycznymi graczami nazywanymi Obrońcą i Atakującym. Obrońca wybiera swoją strategię jako pierwszy, a następnie Atakujący, znając strategię Obrońcy, decyduje się na wybór własnej, co stawia go w uprzywilejowanej sytuacji. Celem gry jest znalezienie równowagi Stackelberga czyli takiej pary strategii Obrońcy i Atakującego, dla których zmiana strategii przez któregokolwiek z graczy pogorszy jego oczekiwany rezultat. Zostało udowodnione, że w przypadku klasy gier rozważanych w rozprawie poszukiwanie tej równowagi jest problemem NP-trudnym.

Model gier obronnych Stackelberga ma istotne znaczenie praktyczne i jest szeroko wykorzystywany w rzeczywistych scenariuszach, takich jak zabezpieczenie lotnisk, walka z kłusownictwem w Afryce czy ochrona statków pasażerskich.

W literaturze dotychczas dominowały podejścia oparte o technikę programowania liniowego i całkowitoliczbowego (MILP), które są nieefektywne pod względem czasu obliczeń i zużywanej pamięci, co czyni je nieskutecznymi w przypadku większych gier.

W rozprawie zaproponowany został algorytm wzorowany na biologicznej ewolucji, który poszukuje optymalnego rezultatu poprzez przetwarzanie populacji potencjalnych rozwiązań. Skuteczność opisanego podejścia została zweryfikowana przez szereg eksperymentów, obejmujących szeroką klasę gier o różnej charakterystyce i poziomie skomplikowania. Otrzymane wyniki pokazały, że zaproponowane podejście w przeważającej większości przypadków zwraca rezultaty optymalne lub bardzo bliskie optymalnym. Jednocześnie charakteryzuje się ono krótszym czasem obliczeń i mniejszym zużyciem pamięci niż metody MILP, co pozwala na rozwiązywanie gier większych i bardziej skomplikowanych. Dodatkowymi atutami zaproponowanego algorytmu są jego uniwersalność oraz łatwość adaptacji do różnych wariantów gier, takich jak gry z częściową

obserwowalnością, gry z ograniczoną racjonalnością Atakującego, czy gry z niepewnością obserwacji.

W rozprawie przeprowadzono wnikliwą analizę działania zaprezentowanego podejścia, przedstawiono wyniki eksperymentów dotyczących różnych wariantów algorytmu i porównanie ich z najlepszymi metodami opisanymi w literaturze. Dodatkowo zaproponowane zostały dwa istotne rozszerzenia bazowej wersji algorytmu wykorzystujące sztuczne sieci neuronowe (podejście neuroewolucyjne) oraz dodatkową rywalizującą populację (podejście koewolucyjne).

Wyniki przeprowadzonych eksperymentów pokazały, że zastosowanie algorytmów ewolucyjnych do poszukiwania równowagi w grach obronnych Stackelberga niesie szereg korzyści i stanowi wartościową alternatywę względem innych stosowanych obecnie metod.

Słowa kluczowe: *algorytmy ewolucyjne, gry obronne, równowaga Stackelberga*

Abstract

Title: *Evolutionary algorithms in sequential Stackelberg Security Games*

The thesis examines the possibility of applying evolutionary algorithms for approximating equilibrium in sequential Stackelberg Security Games. Security Games are played by two non-symmetrical players called the Defender and the Attacker. The Defender commits to his/her strategy first and after that, the Attacker chooses his/her strategy knowing the opponent's strategy. The goal is to find Stackelberg equilibrium which is the pair of the players' strategies that fulfill the following condition: changing strategy by any player will lead to his/her result deflation. It was proven that for the class of games considered in this thesis finding Stackelberg equilibrium is an NP-hard problem.

Stackelberg Security Games have significant practical relevance and they are widely implemented in real-world applications, for instance for airports surveillance, poaching prevention in Africa, or ferries protection in Boston Harbour.

The most common approaches in the literature base on Mixed Integer Linear Programming (MILP). However, they are ineffective in terms of computational complexity and memory consumption which makes them insufficient for larger games.

This thesis proposes a novel algorithm based on biological evolution which explores the results space by maintaining a population of candidate solutions. The effectiveness of the described approach is verified in a series of experiments that cover a wide class of test games with various characteristics and difficulty levels. Obtained results show that the proposed method, in the vast majority of the cases, returns solutions that are optimal or very close to optimal. At the same time, the method demonstrates better computation time scalability and lower memory consumption which allows solving bigger and more complicated games. Another advantage of the proposed algorithm

is its generality and ease of adaptation to various game variants such as games with limited observability, games with bounded rationality of the Attacker, or games with observational uncertainty.

The thesis presents a deep analysis of the proposed approach, comprehensive experimental evaluation of various algorithm modifications and comparison with state-of-the-art methods from the literature. Moreover, two extensions to the baseline algorithm are proposed: artificial neural network utilization (neuroevolutionary approach) and competing population incorporation (coevolutionary approach).

Demonstrated experimental results show that the application of evolutionary algorithms to Stackelberg Security Games offers a number of advantages and is a viable alternative to other currently used methods.

Key words: *evolutionary algorithms, Security Games, Stackelberg Equilibrium*

Podziękowania

Składam serdeczne podziękowania osobom, które w sposób bezpośredni lub pośredni przyczyniły się do powstania rozprawy:

- **Prof. dr hab. Jackowi Mańdziukowi** promotorowi rozprawy za zainteresowanie tematem, okazaną pomoc, cenne uwagi i sugestie, które wpłynęły na ostateczny kształt rozprawy, a także za owocną współpracę przy redakcji artykułów naukowych,
- **Żonie** za codzienne wsparcie, motywację do realizacji ambitnych celów oraz namowę do podjęcia studiów doktoranckich,
- **Rodzicom i Siostrze** za bezwarunkowe wsparcie w każdej sytuacji oraz ukształtowanie osobowości,
- **Kolegom**, a w szczególności uczestnikom Seminarium Inteligencji Obliczeniowej na Wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej, za inspirujące rozmowy i ciekawe pytania.

Spis treści

Rozdział 1

Wprowadzenie

1.1 Zakres rozprawy

Rozprawa doktorska dotyczy zastosowania algorytmów ewolucyjnych do poszukiwania aproksymacji stanu równowagi w wielokrokowych grach obronnych Stackelberga (ang. *Stackelberg Security Games*) [70]. Podstawowym założeniem rozważanych gier jest asymetria graczy. W grze uczestniczy dwóch graczy: Obrońca (w grach Stackelberga nazywany również Liderem (ang. *Leader*)) oraz Atakujący (nazywany też Naśladowcą (ang. *Follower*)). Obrońca wybiera swoją strategię jako pierwszy. Atakujący, znając strategię Obrońcy, decyduje się na wybór własnej, co stawia go w uprzywilejowanej sytuacji. W praktyce Obrońca wybiera strategię mieszaną, która określa prawdopodobieństwa, z jakimi decyduje się on na konkretną postać strategii (konkretne akcje). Atakujący zna ten rozkład prawdopodobieństwa, lecz nie posiada wiedzy odnośnie szczegółowej realizacji strategii Obrońcy w danej rozgrywce. Sytuacja, w której zmiana przyjętej strategii przez któregokolwiek z graczy zmniejszy jego wypłatę (wynik gry) nazywana jest stanem równowagi Stackelberga [42]. Celem gry jest znalezienie pary strategii, które tworzą stan równowagi Stackelberga.

Znalezienie optymalnego rozwiązania (równowagi Stackelberga) jest problemem NP-trudnym [17]. W wielu praktycznych zastosowaniach gier Stackelberga wymagane jest znalezienie strategii w sytuacjach, których poziom skomplikowania uniemożliwia dokładne wyliczenie optymalnego rozwiązania. Powstaje zatem potrzeba zastosowania metod aproksymacyjnych, które pozwolą na szybkie znalezienie strategii bliskich optymalnym. Zaproponowane w tej rozprawie metody są odpowiedzią na tę potrzebę.

1.2 Cele badawcze

Dotychczasowe rozwiązania w obszarze gier obronnych Stackelberga skupione były wokół znajdowania rozwiązań dokładnych przy pomocy programowania liniowego i całkowitoliczbowego [8, 13, 7]. Jednak mają one ograniczony zakres zastosowań ze względu na czasochłonność obliczeń i wysokie zużycie pamięci. Powoduje to naturalne ograniczenie rozmiaru możliwych do rozwiązania problemów, które często nie odpowiadają rzeczywistym potrzebom. Ponadto, zazwyczaj są to metody o ograniczonych zastosowaniach, np. rozwiązujące jedynie wybraną klasę gier (np. gry jednokrokowe o sumie zerowej) lub działające pod pewnymi warunkami. W literaturze istnieje bardzo mało metod, które pozwalałyby na efektywne przybliżone znajdowanie rozwiązań dla większych wielokrokowych gier obronnych Stackelberga [33, 80] lub uniwersalnych algorytmów, które można zastosować do wielu różnych rodzajów gier bez wprowadzania istotnych modyfikacji.

Jednym z niezbadanych obszarów w dziedzinie gier obronnych Stackelberga jest zastosowanie algorytmów ewolucyjnych [15]. Algorytmy ewolucyjne to grupa metod inspirowanych biologicznym procesem ewolucji. Ich głównym obszarem zastosowań są problemy optymalizacyjne, dla których znalezienie dokładnego rozwiązania jest zbyt kosztowne. Naturalnym zatem wydaje się próba ich wykorzystania do problemu znajdowania równowagi Stackelberga, który można potraktować jako szczególny rodzaj dwupoziomowego problemu optymalizacyjnego [16].

Celem rozprawy doktorskiej jest zbadanie możliwości zastosowania metod ewolucyjnych do poszukiwania efektywnych strategii w grach obronnych Stackelberga. Zadaniem jest stworzenie algorytmu, który pozwoli na znajdowanie rozwiązań dla gier o większych rozmiarach oraz będzie możliwy do zastosowania bez istotnych zmian do szerokiej klasy problemów.

Założone cechy projektowanego algorytmu to:

- **skuteczność** - bardzo dobra jakość wyników, powtarzalne uzyskiwanie rezultatów bliskich strategii optymalnej,

- **uniwersalność** - możliwość zastosowania bez istotnych modyfikacji do wielu wariantów gier,

- **efektywność obliczeniowa i pamięciowa** - krótszy (w porównaniu do istniejących rozwiązań) czas działania i mniejsze zużycie pamięci.

1.3 Hipoteza badawcza

W rozprawie zweryfikowana zostanie następująca hipoteza badawcza:
Możliwe jest zastosowanie algorytmów ewolucyjnych do efektywnej aproksymacji strategii w stanie równowagi, w wielokrokowych grach obronnych Stackelberga.

1.4 Opublikowane wyniki

Niektóre z wyników przedstawionych w rozprawie zostały wcześniej opublikowane przez autora i promotora rozprawy w postaci szeregu artykułów w recenzowanych materiałach konferencyjnych.

1. Adam Żychowski, Jacek Mańdziuk. **A generic metaheuristic approach to sequential security games.** *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, strony 2089–2091, 2020 [87], (CORE A*).

   W artykule przedstawiony został generyczny schemat algorytmu ewolucyjnego rozwiązującego gry obronne Stackelberga nazwany EASG. EASG dzięki ogólnemu sformułowaniu umożliwia łatwą adaptację i zastosowanie do szerokiej klasy problemów. Artykuł prezentuje opis podstawowych operatorów ewolucyjnych oraz wstępne wyniki eksperymentalne na 3 rodzajach gier. Zaproponowany algorytm uzyskuje wyniki zbliżone do optymalnych w czasie znacznie krótszym niż metody dokładne. Otrzymane rezultaty pokazują, że zastosowanie podejścia ewolucyjnego w grach obronnych Stackelberga przynosi szereg korzyści i jest obiecującym kierunkiem dalszych badań.

2. Adam Żychowski, Jacek Mańdziuk. **Evolution of strategies in sequential security games**. *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, strony 1434–1442, 2021 [88], (CORE A*).

   Artykuł stanowi istotne rozszerzenie poprzedniej pracy. Przedstawia dokładną wieloaspektową analizę działania proponowanego algorytmu EASG, wprowadza szereg modyfikacji bazowej metody oraz prezentuje dodatkowe wyniki eksperymentów. Rezultaty potwierdzają skuteczność metody EASG. Jej skalowalność czasowa jest znacznie lepsza niż konkurencyjnych metod opisywanych w literaturze. Jednocześnie zaproponowany algorytm ewolucyjny zachowuje dobrą jakość zwracanych rozwiązań, które są bliskie optymalnym.

3. Jan Karwowski, Jacek Mańdziuk, Adam Żychowski, Filip Grajek, Bo An. **A memetic approach for sequential security games on a plane with moving targets**. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, wolumen 33, strony 970–977, 2019 [37], (CORE A*).

   Artykuł wprowadza nową klasę wielokrokowych gier obronnych Stackelberga, które rozgrywane są na płaszczyźnie. Inspiracją do powstania tego modelu gier był praktyczny scenariusz ochrony promów turystycznych. Istotną innowacją, a zarazem trudnością tych gier jest wprowadzenie ruchomych celów oraz ciągłej przestrzeni poszukiwań, co znacząco zwiększa liczbę możliwych do wyboru strategii graczy. Takie rozszerzenie powoduje brak możliwości zastosowania dotychczas proponowanych metod bez ich istotnej modyfikacji. W pracy, oprócz definicji nowych gier, zaprezentowano również kilka metod je rozwiązujących. Przeprowadzone eksperymenty wykazały, że najlepsze rezultaty otrzymywane są przy użyciu wariantu algorytmu ewolucyjnego EASG dostosowanego do rozważanego rodzaju gier. Wyniki potwierdziły, że metoda EASG może być z powodzeniem stosowana również do gier na płaszczyźnie obronnych Stackelberga z ciągłą przestrzenią strategii Obrońcy.

4. Jan Karwowski, Jacek Mańdziuk, Adam Żychowski. **Anchoring theory in sequential Stackelberg games**. *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, strony 1881–1883, 2020 [36], (CORE A*).

   Praca ta stanowi kolejny przykład udanego zastosowania podejścia opartego o metody ewolucyjne. W artykule poruszone zostało zagadnienie niepełnej racjonalności graczy. Na przykładzie jednego z modeli niepełnej racjonalności (teorii zakotwiczenia) zaprezentowano, w jaki sposób bazowy algorytm ewolucyjny EASG może być rozszerzony o ten model i jakie mogą z tego płynąć korzyści. Wyniki wykazały, że uwzględnienie (podczas wyboru strategii Obrońcy) zachowania przeciwnika, które może być nie w pełni racjonalne (wybór nieoptymalnej strategii), umożliwia uzyskanie lepszych wyników. Po raz kolejny podejście ewolucyjne okazało się być najłatwiejsze w adaptacji (do dodatkowego kryterium niepełnej racjonalności Atakującego) oraz najskuteczniejsze pod względem czasu działania.

5. Adam Żychowski, Jacek Mańdziuk. **Learning attacker's bounded rationality model in security games**. *Proceedings of the 19th International Conference on Neural Information Processing*, strony 530–539. Springer, 2021 [89], (CORE A).

   Artykuł prezentuje neuroewolucyjne podejście do problemu poszukiwania równowagi Stackelberga w grach obronnych. Jest to rozszerzenie przedstawionego wcześniej algorytmu ewolucyjnego EASG o sztuczną sieć neuronową (perceptron wielowarstwowy), która pozwala na szybką estymację oczekiwanej wypłaty Obrońcy na podstawie jego strategii oraz definicji gry. Sieć neuronowa zastępuje najbardziej kosztowny obliczeniowo krok algorytmu ewolucyjnego - ewaluację potencjalnych rozwiązań. W ten sposób, przy niewielkim spadku jakości rozwiązań, udało się znacząco przyspieszyć działanie algorytmu. Dodatkową zaletą takiego podejścia jest możliwość znajdowania strategii na podstawie historycznych rozgrywek, bez pełnej wiedzy o przeciwniku (np. postrzeganiu przez niego atrakcyjności ataku poszczególnych celów czy stopniu racjonalności podejmowania decyzji), co jest o wiele bliższe sytuacjom występującym w rzeczywistości.

6. Adam Żychowski, Jacek Mańdziuk, Elizabeth Bondi, Aravind Venugopal, Milind Tambe, Balaraman Ravindran. **Evolutionary approach to security games with signaling**. *International Joint Conference on Artificial Intelligence* (przyjęta do druku). 2022 [91], (CORE A*).

   W artykule zaprezentowano zastosowanie algorytmu ewolucyjnego do klasy gier inspirowanych walką z kłusownictwem w Afryce - gier z sygnalizacją. Ich cechą charakterystyczną jest dostępność dwóch rodzajów zasobów Obrońcy o odmiennej naturze: strażników i dronów, które monitorują obszar gry i mogą się ze sobą komunikować. Gry te uwzględniają również niepewność wynikającą z niedoskonałości sensorów oraz ludzkiej percepcji. W pracy zaproponowano algorytm ewolucyjny dostosowany do charakterystyki poruszanego problemu. Eksperymentalnie pokazano, że szybkość działania i skuteczność zaprezentowanej metody przewyższa konkurencyjne podejścia.

7. Adam Żychowski, Jacek Mańdziuk. **Coevolutionary approach to sequential Stackelberg security games**. *International Conference on Computational Science* (przyjęta do druku). 2022 [90], (CORE A).

   Artykuł przedstawia nowe koewolucyjne podejście do poszukiwania równowagi Stackelberga w grach obronnych. Działanie metody oparte jest o dwie konkurujące ze sobą populacje zawierające strategie graczy - Obrońcy i Atakującego. Rozwijane naprzemiennie w procesie ewolucji pozwalają na szybką weryfikację jakości strategii przeciwnika, co pozwala uniknąć najbardziej czasochłonnej procedury ewaluacji używanej w bazowym algorytmie EASG opisywanym w poprzednich pracach. Takie podejście przyspieszyło proces poszukiwania rozwiązania przy znikomej utracie jakości wyników.

8. Jan Karwowski, Jacek Mańdziuk, Adam Żychowski. **Sequential Stackelberg games with bounded rationality**, (w recenzji w czasopiśmie).

   W pracy zaprezentowano rozszerzenie jednego z modeli ograniczonej racjonalności, teorii zakotwiczenia, do wielokrokowych gier obronnych Stackelberga. Modyfikacja sformułowania tej teorii umożliwiła jej wdrożenie do metod dokładnych

poszukiwania równowagi Stackelberga opartych o technikę programowania liniowego i całkowitoliczbowego. Eksperymenty przeprowadzone z udziałem ludzi pokazały skuteczność zaproponowanego podejścia oraz potwierdziły zasadność jego stosowania. Dodatkowo w pracy przedstawiono implementację teorii zakotwiczenia do dwóch metod przybliżonych: algorytmu ewolucyjnego (EASG) [88] oraz przeszukiwania drzew metodą Monte Carlo (O2UCT) [35].

1.5 Układ pracy

Rozprawa podzielona jest na 10 rozdziałów. Rozdział 1 stanowi wprowadzenie prezentujące tematykę rozprawy, jej zakres oraz sformułowanie hipotezy badawczej. W rozdziale 2 znajduje się wstęp teoretyczny oraz omówienie poruszanych zagadnień z zakresu teorii gier, problemów optymalizacyjnych i algorytmów ewolucyjnych. Rozdział 3 definiuje równowagę Stackelberga oraz gry obronne z przytoczeniem szeregu ich praktycznych zastosowań. Kolejny rozdział poświęcony jest przeglądowi istniejących w literaturze metod poszukiwania równowagi Stackelberga. Rozdział 5 przedstawia ogólny schemat proponowanego rozwiązania opartego o procesy ewolucyjne. Wyniki eksperymentów weryfikujących jakość zaprezentowanego algorytmu ewolucyjnego na grach o różnej charakterystyce zostały zademonstrowane w rozdziale 6. Następna część zawiera dokładniejszą analizę zaproponowanego algorytmu pod kątem wrażliwości na zmiany poszczególnych parametrów, jego zbieżności oraz stabilności. W tej części zostały omówione dodatkowe modyfikacje algorytmu. Badaniu zagadnienia ograniczonej racjonalności w grach obronnych Stackelberga został poświęcony rozdział 8. Zaproponowano w nim rozszerzenie opisywanej metody o sztuczną sieć neuronową oceniającą jakość weryfikowanych w procesie ewolucji rozwiązań. Rozdział 9 prezentuje nowe, koewolucyjne podejście do badanego zagadnienia oraz ocenę jego skuteczności. Ostatnia część rozprawy stanowi podsumowanie przeprowadzonych badań, przedstawia wnioski oraz możliwości dalszego rozwoju zaprezentowanych prac badawczych.

Rozdział 2

Definicje

W rozdziale tym przedstawione zostaną definicje podstawowych zagadnień z obszaru teorii gier, teorii optymalizacji oraz algorytmów ewolucyjnych, które używane będą w dalszej części rozprawy.

2.1 Podstawowe pojęcia teorii gier

Podstawowym pojęciem teorii gier jest gra. Wyodrębnić można różne rodzaje gier [23]: kooperacyjne/niekooperacyjne, w postaci normalnej/ekstensywnej, z pełną/niepełną informacją itd. Najprostszym i podstawowym rodzajem gier są gry w postaci normalnej (inaczej gry strategiczne [63]).

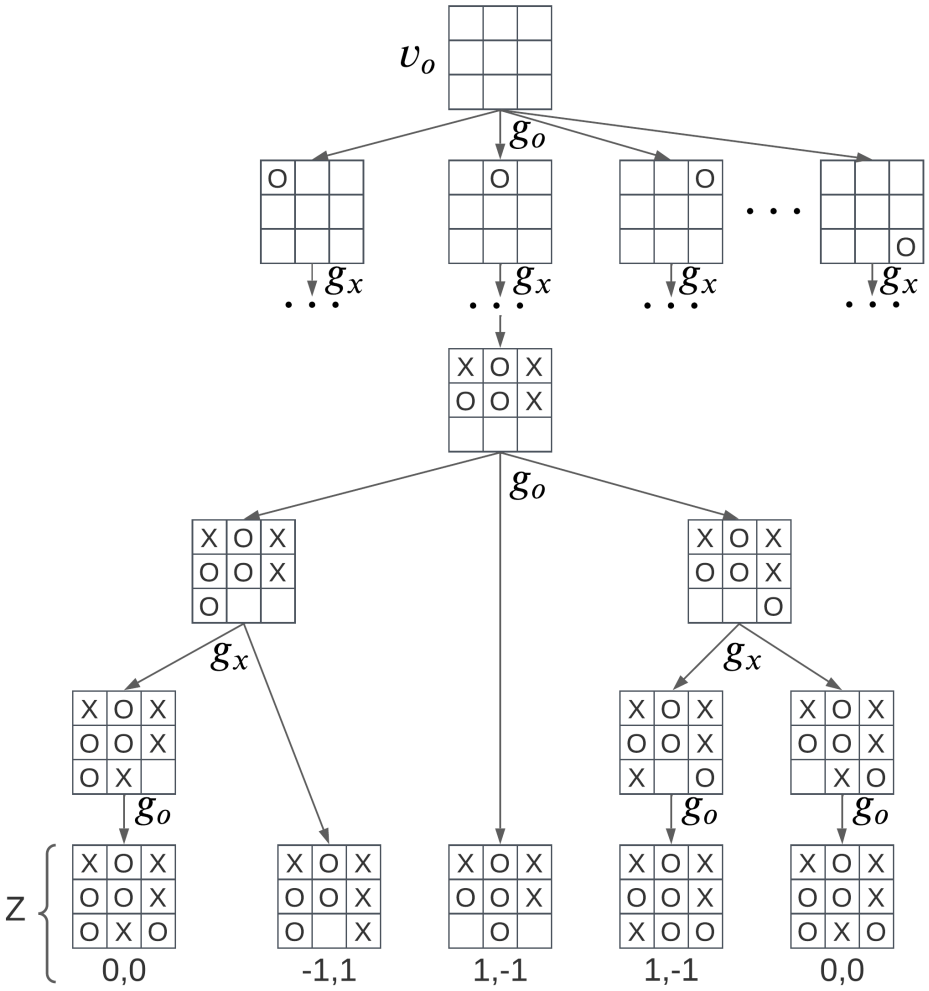
Definicja 1. ***Gra w postaci normalnej*** *to trójka* $\langle G, (A_g)_{g\in G}, (u_g)_{g\in G}\rangle$*, gdzie* G *jest zbiorem graczy,* A_g *jest zbiorem akcji (możliwych ruchów) gracza* g*,* $u_g : A \to \mathbb{R}$ *jest funkcją przypisującą wypłatę graczowi* g *w zależności od wybranych przez graczy akcji* $(A = \times A_g, g \in G)$*.*

Powyższa definicja zakłada, że każdy z graczy wybiera jedną akcję, która jest wykonywana jednocześnie z akcjami pozostałych graczy, a wynikiem jest wypłata przypisana każdemu z nich. W przypadku, gdy kolejność podejmowanych akcji nie jest jednoczesna (gracze wykonują ruchy sekwencyjnie) wprowadza się pojęcie gier w formie ekstensywnej [65]. Zazwyczaj takie gry przedstawiane są w postaci drzewa [23].

Definicja 2. ***Gra w postaci ekstensywnej*** *to tupla* $\langle G, D, v_0, Z, A, f, a, (u_g)_{g\in G}\rangle$*, gdzie*
G *jest zbiorem graczy,*
D *jest drzewem gry - grafem z określonymi wierzchołkami* (D_V) *(nazywanymi stanami*

gry) i połączeniami między nimi (D_E),
$v_0 \in D_V$ jest wyróżnionym wierzchołkiem - korzeniem drzewa (stanem początkowym),
$Z \subset D_V$ jest zbiorem stanów terminalnych (podzbiorem wierzchołków D_V),
A jest zbiorem wszystkich akcji,
$a : D_E \to A$ jest funkcją przypisującą każdej krawędzi drzewa akcję,
$f : D_V \setminus Z \to G$ jest funkcją przypisującą każdemu nieterminalnemu stanowi gracza, który w danym stanie ma wykonać akcję (polegającą na wyborze kolejnego wierzchołka połączonego z aktualnym, czyli przejściu do kolejnego stanu),
$u_g : Z \to \mathbb{R}$ jest funkcją przypisującą wypłatę graczowi g w zależności od stanu terminalnego.

Rysunek 2.1 przedstawia przykład gry w formie ekstensywnej.

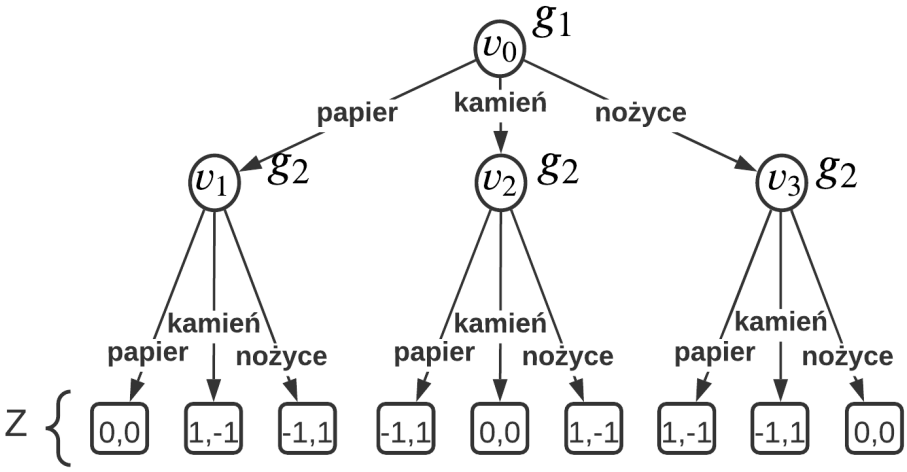


Rysunek 2.1: Fragment drzewa gry "kółko i krzyżyk" w formie ekstensywnej. W stanie początkowym v_0 gracz g_o podejmuje akcję. W wierzchołkach będącym wynikiem akcji gracza g_o swoją akcję podejmuje gracz g_x. Ostatecznie gra kończy się w jednym ze stanów terminalnych Z, a każdy z graczy otrzymuje odpowiednią wypłatę zaprezentowaną na dole (pierwsza liczba oznacza wypłatę gracza g_o, druga g_x).

W przypadku gier, w których gracze nie mają pełnej informacji o podejmowanych przez przeciwnika akcjach (gry z niepełną informacją), ważnym pojęciem jest zbiór informacyjny.

Definicja 3. ***Zbiór informacyjny*** *(ang. information set) to kolekcja wierzchołków drzewa gry $I \subseteq D_V$ spełniających następujące warunki:*
1) $\exists_{g \in G} \forall_{v \in I} f(v) = g$
2) $\forall_{v_1, v_2 \in I} A_{v_1} = A_{v_2}$*, gdzie* A_v *oznacza zbiór dostępnych akcji w wierzchołku* v*.*
Innymi słowy we wszystkich wierzchołkach danego zbioru informacyjnego ten sam gracz podejmuje akcję oraz dostępne akcje w każdym z tych wierzchołków są takie same.

Zbiory informacyjne określają jaką wiedzę o ruchach przeciwnika mają gracze. Dzięki pojęciu zbiorów informacyjnych, gry w postaci normalnej można przedstawić jako gry w formie ekstensywnej.



Rysunek 2.2: Przykład gry papier-kamień-nożyce w formie ekstensywnej między graczami g_1 i g_2. Gra kończy się w jednym ze stanów terminalnych Z, a każdy z graczy otrzymuje odpowiednią wypłatę (pierwsza liczba oznacza wypłatę gracza g_1, druga g_2).

Przykładem może być gra "papier-kamień-nożyce", której reprezentacja została przedstawiona na rysunku 2.2. Jeśli przyjmiemy, że wierzchołki v_1, v_2, v_3 tej gry będą należały do różnych zbiorów informacyjnych, to będzie oznaczało, że te stany są dla gracza g_2 rozróżnialne i zna on decyzję podjętą przez gracza g_1. Jeśli natomiast przyjmiemy, że wszystkie te wierzchołki należą do jednego zbioru informacyjnego, to gracz g_2 będzie wybierał akcję bez wiedzy, w którym konkretnie wierzchołku gry aktualnie się znajduje, tj. nie wie, jaką decyzję podjął gracz g_1, co jest najczęściej przyjmowaną zasadą gry "papier-kamień-nożyce".

Definicja 4. ***Strategia*** *gracza g jest zdefiniowana jako funkcja $\sigma_g : \mathcal{I} \to A$, która do każdego zbioru informacyjnego przypisuje akcję, która zostanie wykonana przez gracza g w tym zbiorze.*

Powyżej zdefiniowana strategia często nazywana jest *strategią prostą* lub *strategią czystą*, aby odróżnić ją od poniżej zdefiniowanej strategii mieszanej.

Definicja 5. ***Strategia mieszana*** *gracza g (oznaczana przez π_g) to rozkład prawdopodobieństwa na zbiorze wszystkich możliwych strategii prostych: $\pi_g : \Sigma_g \to \mathbb{R}, \int_{\Sigma_g} \pi_g = 1$, gdzie Σ_g jest zbiorem wszystkich strategii prostych gracza g.*

Strategia mieszana oznacza, że gracz najpierw zgodnie z zadanym rozkładem prawdopodobieństwa wybiera jedną strategię prostą, a następnie wszystkie decyzje podejmuje według tej strategii.

Definicja 6. ***Profil strategii*** *to zbiór strategii wszystkich graczy (po jednej dla każdego z nich): $p = \{s_g : g \in G\}$.*

W profilu mogą występować zarówno strategie proste jak i mieszane.

Można zauważyć, że dany profil strategii prostych jednoznacznie wyznacza stan terminalny gry, bowiem poszczególne strategie definiują akcje wykonywane przez graczy. Przez $u_g(p)$ będziemy oznaczali wypłatę gracza g dla profilu strategii p. W przypadku, gdy w profilu występuje co najmniej jedna strategia mieszana będzie to wartość oczekiwana wypłaty.

Definicja 7. *Strategia s_g^** ***dominuje*** *strategię s_g wtedy i tylko wtedy, gdy*

$$\forall_{p \in P_{-g}} \; u_g(s_g^* \cup p) \geqslant u_g(s_g \cup p)$$

oraz

$$\exists_{p^* \in P_{-g}} \; u_g(s_g^* \cup p^*) > u_g(s_g \cup p^*)$$

gdzie P_{-g} oznacza zbiór wszystkich profili strategii bez strategii gracza g.

Definicja 8. *Strategia s_g^* jest* ***strategią dominującą*** *gracza g wtedy i tylko wtedy, gdy s_g^* dominuje wszystkie inne strategie gracza g.*

Definicja 9. ***Równowaga*** *to profil strategii $p^* = \{s_g^* : g \in G\}$, w którym wszystkie strategie są dominujące.*
Innymi słowy jest to zbiór strategii poszczególnych graczy, dla których przy pewnych założeniach (określonych w zasadach konkretnej gry), żadnemu z graczy nie opłaca się zmieniać wybranej strategii.

W rozprawie są rozważane gry sekwencyjne w postaci ekstensywnej z dwoma graczami. Poszukiwana jest para strategii mieszanych graczy, składająca się na pewien szczególny rodzaj równowagi - równowagę Stackelberga, która zostanie zdefiniowana w dalszej części pracy.

2.2 Problemy optymalizacyjne

Problemy optymalizacyjne to jedne z najczęściej spotykanych modeli opisujących wiele praktycznych zagadnień. Ich formalna definicja jest następująca.

Definicja 10. ***Problem optymalizacyjny*** *[82] zdefiniowany jest przez trzy elementy: przestrzeń rozwiązań* Ω*, funkcję oceny rozwiązań* $f : \Omega \rightarrow \mathbb{R}^n$ *oraz relację* $\succeq$ *określoną na elementach przeciwdziedziny funkcji* f*.*

Funkcję f nazywa się również *funkcją kosztu* lub *funkcją celu*.

W niniejszej rozprawie rozważane będą jedynie problemy jednokryterialne, czyli takie, dla których przeciwdziedzina funkcji f jest podzbiorem liczb rzeczywistych, tj. $f : \Omega \rightarrow \mathbb{R}$. Dodatkowo będą to problemy maksymalizacji, czyli relacja $\succeq$ będzie klasyczną relacją większości ($\geqslant$) zdefiniowaną na zbiorze liczb rzeczywistych.

Problemy optymalizacyjne możemy podzielić ze względu na charakterystykę przestrzeni rozwiązań na problemy ciągłe i dyskretne.

Definicja 11. ***Problem ciągły*** *[82] to problem, którego rozwiązania są reprezentowane przez wektory liczb rzeczywistych -* $\Omega \subseteq \mathbb{R}^n$*.*

Definicja 12. ***Problem dyskretny*** *[82] to problem, którego rozwiązania są reprezentowane przez wektory liczb całkowitych -* $\Omega \subseteq \mathbb{Z}^n$*.*

Szczególnym rodzajem problemów optymalizacyjnych są problemy optymalizacyjne dwupoziomowe [16]. Zalicza się do nich problem znajdowania równowagi Stackelberga, będący głównym tematem rozprawy. Problemy dwupoziomowe są złożeniem dwóch zagnieżdżonych problemów optymalizacyjnych. Ocena konkretnego rozwiązania x z przestrzeni Ω_1 jest sama w sobie dodatkowym problemem optymalizacyjnym i wymaga znalezienia optymalnego rozwiązania y z przestrzeni Ω_2. Problemy te nazywane są odpowiednio zewnętrznym i wewnętrznym. Definicja 13 przedstawia sformułowanie problemu dwupoziomowego w przypadku maksymalizacji.

Definicja 13. ***Dwupoziomowy problem optymalizacyjny*** *(maksymalizacyjny) można zdefiniować jako:*

$$\max_{x\in\Omega_1, y\in R} F(x,y),$$

$$R = \{y\colon y = \arg\max_{z\in\Omega_2} f(x,z)\},$$

gdzie F jest funkcją oceny rozwiązań problemu zewnętrznego, f jest funkcją oceny rozwiązań problemu wewnętrznego, natomiast $x \in \Omega_1$ oraz $y \in \Omega_2$ to odpowiednio elementy przestrzeni rozwiązań tych problemów.

2.3 Algorytmy ewolucyjne

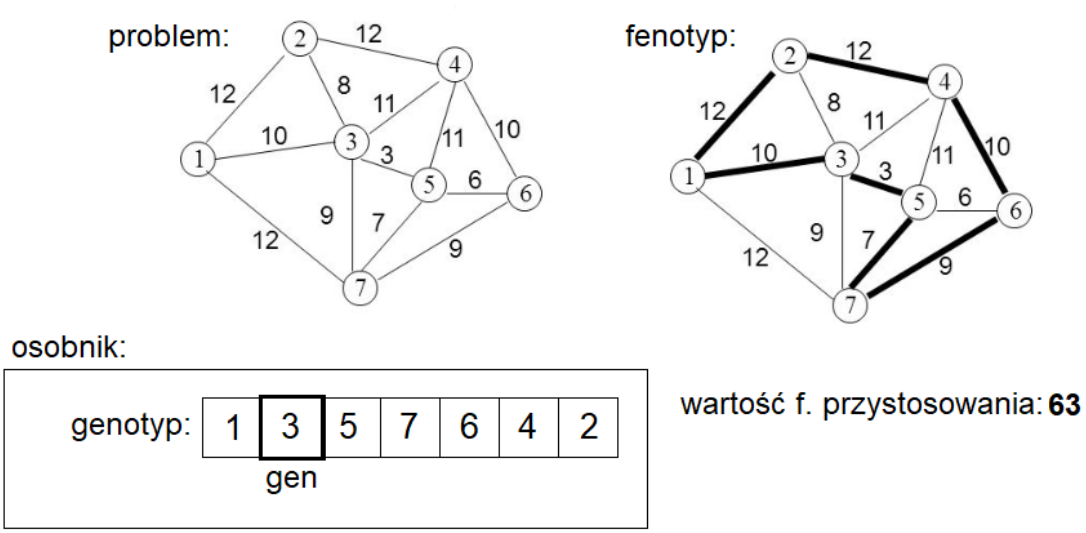
Algorytmy ewolucyjne to rodzaj metaheurystycznych algorytmów optymalizacyjnych inspirowanych obserwacją działania biologicznej ewolucji. W związku z tym wiele pojęć takich jak na przykład gen, chromosom, fenotyp, pochodzących z nauk biologicznych, występuje również w algorytmach ewolucyjnych.

2.3.1 Podstawowe pojęcia

Osobnik to podstawowa jednostka podlegająca ewolucji. Zbiór osobników tworzy **populację** (ozn. $\mathcal{P}$). Populacja ma stałą określoną wielkość $|\mathcal{P}| = N$. Każdy osobnik zawiera kompletny i jednoznaczny opis potencjalnego rozwiązania (elementu przestrzeni poszukiwań Ω), który nazywany jest **genotypem**. Miejscem przechowywania genotypu jest **chromosom**. Genotyp składa się z **genów**, czyli pojedynczych cech, właściwości zakodowanego rozwiązania. Natomiast **fenotyp** to ujawniające się na zewnątrz cechy danego osobnika, czyli rozkodowane parametry rozwiązania, które podlegają ocenie. Rozwiązania oceniane są za pomocą tzw. **funkcji przystosowania** (zwanej też funkcją dopasowania lub funkcją celu). Określa ona jakość rozwiązania reprezentowanego przez danego osobnika.

W celu zobrazowania wyżej wymienionych pojęć posłużmy się przykładem dla problemu komiwojażera przestawionym na rysunku 2.3. Problem komiwojażera polega na znalezieniu cyklu Hamiltona o najmniejszej wadze w grafie spójnym. Jednym z możliwych sposobów kodowania rozwiązania jest jego zapis w postaci wektora identyfikatorów kolejnych wierzchołków cyklu, jak to zostało przedstawione w przykładzie. Wtedy genem będzie pojedynczy element tego wektora - identyfikator wierzchołka, a

genotypem cały wektor. Będzie to jednocześnie chromosom tego osobnika. Natomiast fenotypem w tym przypadku będzie cykl w grafie, który odpowiada zakodowanemu rozwiązaniu. Wartością funkcji przystosowania jest suma wag krawędzi tego cyklu.



Rysunek 2.3: Przykład kodowania rozwiązań dla problemu komiwojażera.

Charakterystycznym elementem algorytmów ewolucyjnych są operacje mutacji, krzyżowania oraz selekcji, których idea również jest inspirowana procesami biologicznymi.

2.3.2 Mutacja

Mutacja jest funkcją $M : \Omega \rightarrow \Omega$ przekształcającą elementy przestrzeni poszukiwań. Najczęściej wprowadza ona pewne losowe zaburzenie zakodowanego rozwiązania. Operacja ta wykonywana jest na genotypie danego osobnika, a w wyniku jej działania powstaje nowe rozwiązanie. Celem mutacji jest zapewnienie różnorodności populacji oraz eksploracja nowych rozwiązań, przeszukiwanie nowych obszarów.

Przykładem mutacji, w przypadku rozwiązań reprezentowanych przez wektor binarny, może być zamiana losowo wybranego bitu na przeciwny lub odwrócenie kolejności fragmentu wektora. W przypadku kodowania rozwiązań przy pomocy liczb rzeczywistych mutacją może być na przykład dodanie szumu zgodnie z rozkładem normalnym ($M = \mathcal{N}(\mu, \sigma)$) czy zamiana dwóch liczb miejscami (w przypadku wektora liczb).

Rodzaj zastosowanej mutacji zwykle zależy od rozwiązywanego problemu. Na przykład dla problemu komiwojażera i kodowania rozwiązań zaprezentowanego na rysun-

ku 2.3 mutacją może być zamiana miejscami dwóch losowo wybranych identyfikatorów wierzchołków[1].

2.3.3 Krzyżowanie

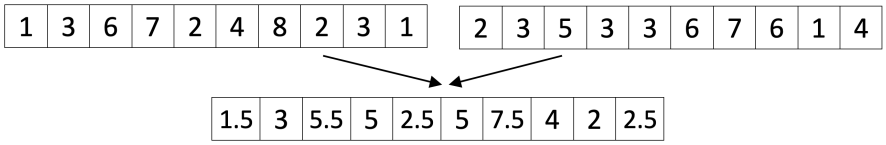
Krzyżowanie jest funkcją $K : \Omega^n \rightarrow \Omega^m$ przekształcającą pewną liczbę (najczęściej $n = 2$) elementów przestrzeni poszukiwań w inne jej elementy (najczęściej $m \in 1, 2$). Krzyżowanie ma na celu przemieszanie zakodowanych w genotypach cech wybranych osobników. Zwykle krzyżowane są 2 osobniki (nazywane rodzicami). Ich cechy ulegają rekombinacji, w wyniku której powstaje jeden lub dwa osobniki potomne. Ideą krzyżowania jest eksploatacja wybranego obszaru przestrzeni poszukiwań i oczekiwanie, że w wyniku połączenia odpowiednich cech osobników-rodziców powstanie lepiej przystosowany osobnik potomny (reprezentujący lepsze rozwiązanie).

Istnieje kilka standardowych modeli krzyżowania. Są to na przykład krzyżowania wymieniające lub uśredniające. **Krzyżowanie wymieniające** polega na wyborze jednego (krzyżowanie jednopunktowe) lub wielu (krzyżowanie wielopunktowe) miejsc podziału chromosomu, a następnie wymianie odpowiednich fragmentów między osobnikami. **Krzyżowanie uśredniające** natomiast przeprowadza określoną operację na kolejnych odpowiadających sobie parach genów w chromosomach rodziców. Tą operacją może być średnia, średnia ważona, operator AND, XOR, itp. Przykłady krzyżowania wymieniającego i uśredniającego zostały zaprezentowane na rysunku 2.4.

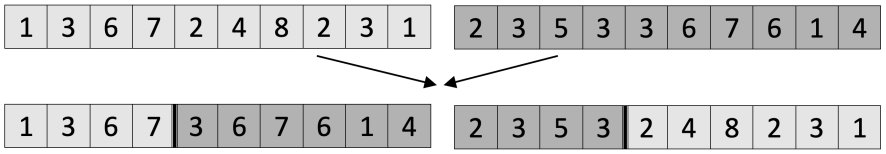
Wybór odpowiednich operatorów mutacji i krzyżowania istotnie zależy od rozwiązywanego problemu oraz od przyjętego sposobu kodowania. Przykładowo, dla przytaczanego wcześniej problemu komiwojażera i zakodowania rozwiązań jak na rysunku 2.3, krzyżowanie uśredniające lub wymieniające może doprowadzić do powstania rozwiązań niedopuszczalnych. Właściwe zaprojektowanie tych operatorów ma kluczowe znaczenie dla działania algorytmu - jego szybkości oraz jakości znajdowanych rozwiązań.

W zależności od wybranego wariantu, nowe osobniki (stworzone w wyniku działania mutacji i/lub krzyżowania) albo zastępują te, z których powstały, albo dołączane są do istniejących już rozwiązań.

[1] Jeśli graf nie jest kliką, to taka mutacja może doprowadzić do powstania rozwiązań niedopuszczalnych. W takim przypadku można powtarzać mutację lub przypisywać takim rozwiązaniom istotnie gorszą wartość funkcji przystosowania.



(a) Krzyżowanie uśredniające.



(b) Krzyżowanie wymieniające.

Rysunek 2.4: Przykłady operatorów krzyżowania.

2.3.4 Ewaluacja

Po zastosowaniu operatorów mutacji i krzyżowania następuje *ewaluacja* nowych osobników, czyli wyliczenie wartości funkcji przystosowania f. Innymi słowy, oznacza to sprawdzenie, jak dobre jest rozwiązanie reprezentowane przez danego osobnika. Często jest to najbardziej kosztowna obliczeniowo część algorytmu ewolucyjnego. W wyniku ewaluacji każdemu osobnikowi przypisywana jest wartość liczbowa, która stanowi podstawę jego oceny.

2.3.5 Selekcja

Procedura selekcji decyduje o tym, które z osobników zostaną wybrane do stworzenia kolejnej generacji, nowego pokolenia. Podstawową ideą działania selekcji jest zasada, że większą szansę na wybór mają osobniki lepiej przystosowane (z wyższą wartością funkcji przystosowania). Zazwyczaj każdy osobnik (nawet najsłabszy) ma niezerową szansę na sukcesję, czyli przejście do kolejnego pokolenia. Selekcja może prowadzić do wyboru tego samego osobnika kilkukrotnie (stworzenia jego kopii).

Jednym z najpopularniejszych rodzajów selekcji jest *selekcja ruletkowa*. Polega ona na wielokrotnym losowaniu osobników zgodnie z rozkładem prawdopodobieństwa proporcjonalnym do wartości funkcji przystosowania, tzn. osobnik o z populacji $\mathcal{P}$ będzie wylosowany z prawdopodobieństwem $\frac{f(o)}{\sum_{o_i \in \mathcal{P}} f(o_i)}$, gdzie $f(x)$ to znormalizowana do przedziału $[0, 1]$ wartość funkcji przystosowania osobnika x. Takie losowanie powtarzane jest wielokrotnie (ze zwracaniem), dopóki nie zostanie osiągnięta docelowa wielkość

populacji w nowym pokoleniu.

Inną często spotykaną procedurą wyboru osobników do kolejnego pokolenia jest *selekcja turniejowa* [4]. Polega ona na wielokrotnym tworzeniu tzw. turniejów, w ramach których osobniki "walczą" o sukcesję. W tym rodzaju selekcji najpierw wybierany jest losowo podzbiór t_n osobników, które będą uczestniczyć w turnieju. Wartość t_n jest ustaloną z góry wielkością turnieju. Następnie spośród uczestników turnieju wybierany jest jeden osobnik do kolejnego pokolenia. Wybór ten może następować w sposób deterministyczny, czyli promowany jest osobnik z najwyższą wartością funkcji przystosowania, lub w sposób losowy, ale proporcjonalny do przystosowania, jak np. poprzez procedurę selekcji ruletkowej opisaną powyżej. Popularnym podejściem jest również wprowadzenie dodatkowego parametru nazywanego *presją selekcji* (p_s), który określa prawdopodobieństwo, z jakim promowany jest osobnik z najwyższym przystosowaniem spośród uczestników turnieju. Pseudokod procedury selekcji turniejowej został przedstawiony jako algorytm 2.1.

Algorytm 2.1: Przykład procedury selekcji turniejowej z presją selekcji p_s.

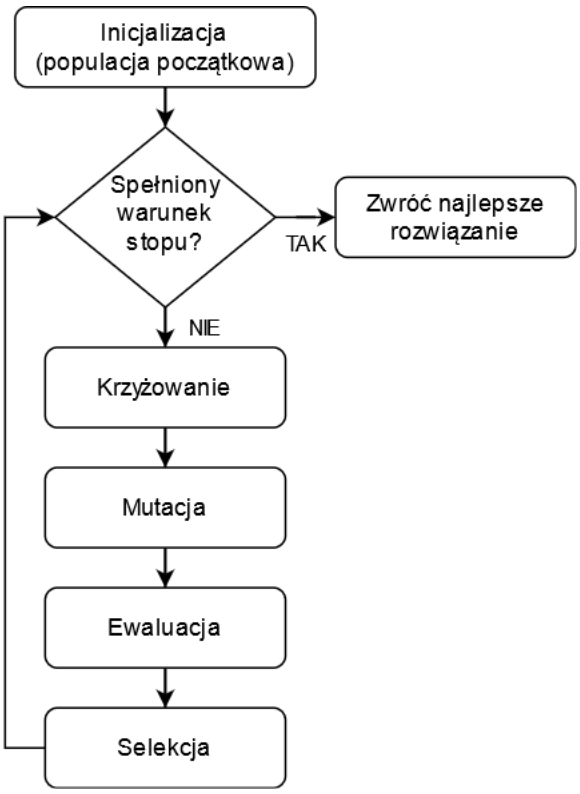
```
1 SelectionTournament (P)
2     P_new ← ∅
3     while |P_new| < N do
4         T ← random(P,t_n)   // losowanie t_n osobników do turnieju T ⊆ P
5     if p_s ⩾ rand([0,1]) then
6         P_new ← P_new ∪ arg max_{x∈T} f(x)
7     else
8         P_new ← P_new ∪ random(T,1)
9     return P_new
```

2.3.6 Elitaryzm

Większość rodzajów selekcji nie daje pewności, że najlepiej przystosowany osobnik zostanie przeniesiony do kolejnego pokolenia i może zdarzyć się tak, że najlepsze rozwiązanie zostanie usunięte z populacji. Z tego powodu częstym podejściem jest zastosowanie *elitaryzmu*. Polega on na tym, że określona liczba najlepiej przystosowanych osobników (przynajmniej jeden, ale zazwyczaj jest ich kilka) jest automatycznie kopiowana do kolejnego pokolenia bez udziału w procedurze selekcji.

2.3.7 Schemat algorytmu ewolucyjnego

Ogólny schemat algorytmu ewolucyjnego został zaprezentowany na rysunku 2.5. W literaturze spotykane są różne warianty kolejności stosowania operatorów ewolucyjnych. W niektórych specyficznych zastosowaniach można spotkać również pominięcie którejś z procedur, np. krzyżowania lub mutacji.



Rysunek 2.5: Ogólny schemat działania algorytmu ewolucyjnego.

Na początku działania algorytmu ewolucyjnego tworzona jest populacja początkowa. Zazwyczaj składa się ona z losowych osobników, które, w miarę możliwości, w sposób równomierny pokrywają przestrzeń rozwiązań. Czasami jednak do wygenerowania populacji początkowej używany jest algorytm heurystyczny, który wskazuje obszary, gdzie znajdują się potencjalnie dobre rozwiązania. Po wygenerowaniu populacji początkowej następuje tworzenie nowych pokoleń poprzez wielokrotne zastosowanie operacji mutacji, krzyżowania, ewaluacji i selekcji. Procedura generowania nowych pokoleń powtarzana jest aż do momentu spełnienia warunku stopu. Warunkiem stopu może być ustalona na początku liczba generacji, brak poprawy najlepszego rozwiązania (stagnacja) czy określony czas działania algorytmu.

Algorytmy 2.2 oraz 2.3 prezentują pseudokody typowego przebiegu algorytmu ewolucyjnego oraz procedury generowania nowego pokolenia.

Algorytm 2.2: Pseudokod algorytmu ewolucyjnego.

```
1 EvolutionaryAlgorithm
2     P ← InitializeRandomPopulation()
3     while stop condition not satisfied do
4         P ← NewGeneration(P)
5     return arg max_{x∈P} f(x)
```

Algorytm 2.3: Procedura generowania nowego pokolenia.

```
1 NewGeneration (P)
2     P ← P ∪ Crossover(P_c)   // P_c ⊆ P - osobniki do krzyżowania
3     P ← P ∪ Mutate(P_m)      // P_m ⊆ P - osobniki podlegające mutacji
4     Evaluate(P)              // obliczenie funkcji przystosowania
5     P_new ← GetElite(P)
6     while |P_new| < N do
7         P_new ← P_new ∪ Select(P)
8     return P_new
```

Bardziej szczegółowy opis oraz więcej informacji na temat algorytmów ewolucyjnych, ich wariantów, wpływu parametrów czy podstaw teoretycznych ich działania można znaleźć w publikacjach [2, 47, 48].

2.3.8 Algorytmy memetyczne

Szczególnym rodzajem algorytmów ewolucyjnych są *algorytmy memetyczne*. Ich nazwa pochodzi od pojęcia "mem" [21] oznaczającego jednostkę informacji przekazywanej w sposób inny niż biologiczny (genetyczny). Najczęściej chodzi o informacje kulturowe, zmiany zachodzące i przekazywane w trakcie życia jednego pokolenia takie jak na przykład język, taniec, moda. Pojęcie algorytmów memetycznych, wprowadzone w 1989 roku [49], oznacza algorytmy, które są połączeniem algorytmu ewolucyjnego z algorytmem poszukiwań lokalnych. Klasyczne algorytmy ewolucyjne są skuteczne w odnajdywaniu obszaru, w którym znajduje się optimum, ale słabiej radzą sobie w odnalezieniu optimum lokalnego wewnątrz tego obszaru.

Algorytmy memetyczne adresują ten problem poprzez dodanie dodatkowej procedury, która przeszukuje otoczenie rozwiązania w celu jego optymalizacji. Idea ta może być zrealizowana między innymi dzięki dodaniu kroku algorytmu, który przed ewaluacją rozwiązań próbuje je zmienić (polepszyć) lub poprzez modyfikację operatorów mutacji i/lub krzyżowania. Przykładowo, w krzyżowaniu wymieniającym może to być dobranie optymalnego punktu przecięcia. Natomiast w przypadku mutacji może polegać to na wyborze do modyfikacji genu, którego zmiana w największym stopniu poprawi rozwiązanie. W rozwiązywaniu problemu komiwojażera taką lokalną optymalizacją może być zastosowanie algorytmu 2-OPT [18], który niskim kosztem pozwala na polepszenie wyniku poprzez wybór odpowiednich krawędzi w grafie.

2.3.9 Algorytmy koewolucyjne

Jednym z wariantów algorytmów ewolucyjnych są *algorytmy wielopopulacyjne*, w których ewolucja przebiega w wielu populacjach równolegle. Jednak populacje te nie działają w sposób w pełni niezależny, lecz co pewną liczbę generacji następuje wymiana informacji między nimi, na przykład poprzez migrację osobników. Jednym ze szczególnych rodzajów algorytmów wielopopulacyjnych są *algorytmy koewolucyjne* inspirowane biologicznymi relacjami między gatunkami w przyrodzie (np. symbioza, pasożytnictwo, drapieżnictwo). Charakterystyczną cechą tych algorytmów jest to, że kierunek ewolucji danej populacji uzależniony jest od innej populacji. Realizacja tej zależności następuje w trakcie wyliczania wartości funkcji przystosowania.

Dwa najczęściej spotykane modele to kooperacja i konkurencja. W kooperacji gatunki są zachęcane do "współpracy" przez nagradzanie za wspólne rozwiązywanie problemów i karane za zbyt dużą samodzielność. Wspólnie tworzą pewne rozwiązanie - oceniane jest rozwiązanie złożone z genotypów osobników z różnych populacji. Ten model ma zastosowanie w problemach zbyt skomplikowanych dla jednej populacji lub zadaniach, w których można łatwo wydzielić komponenty.

W modelu konkurencji następuje swoisty "wyścig zbrojeń" populacji - polepszenie jednej populacji powoduje zmniejszenie średniej funkcji przystosowania osobników innej populacji. Naturalnym przykładem może być ewolucja strategii graczy w niekooperacyjnych grach o sumie zerowej (np. warcaby, go) - każda z populacji reprezentuje

strategię jednego z graczy, im lepsza jest strategia danego gracza, tym mniej skuteczna jest strategia pozostałych.

Dokładniejszy opis algorytmów koewolucyjnych, ich wariantów, podstaw działania oraz zastosowań można znaleźć w [64, 27].

Rozdział 3

Gry obronne Stackelberga

W grach obronnych Stackelberga rozważanych w rozprawie wyróżniamy dwóch graczy[1]: Obrońcę (O) i Atakującego (A). Ich wiedza o strategii przeciwnika jest niesymetryczna. Obrońca wybiera swoją strategię jako pierwszy, a następnie Atakujący, znając wybór Obrońcy, decyduje o swojej strategii. Uwidacznia się więc pewna przewaga Atakującego, który posiada więcej informacji. Obrońca jednak nie musi wybierać strategii prostej, zazwyczaj jest to strategia mieszana. Oznacza to, że Atakujący zna jedynie rozkład prawdopodobieństwa, z jakimi przeciwnik wybierze konkretną strategię prostą, nie zna natomiast realizacji, czyli konkretnych akcji, jakie podejmie Obrońca.

3.1 Równowaga Stackelberga

Parę strategii Atakującego i Obrońcy, dla których zmiana strategii któregokolwiek z graczy spowoduje zmniejszenie jego wypłaty nazywamy *równowagą Stackelberga*.

Definicja 1. ***Równowaga Stackelberga*** *to para strategii* (π_O^*, π_A^*) *będących rozwiązaniem poniższego układu równań:*

$$\begin{cases} \pi_O^* = \arg\max_{\pi_O \in \Pi_O} u_O\left(\pi_O, R(\pi_O)\right) \\ \pi_A^* = R(\pi_O^*) \\ R(\pi_O) = \arg\max_{\pi_A \in \Pi_A} u_A(\pi_O, \pi_A) \end{cases} \tag{3.1}$$

gdzie π_O *oraz* π_A *oznaczają odpowiednio strategię mieszaną Obrońcy i Atakującego,* Π_O *i* Π_A *są zbiorami wszystkich możliwych strategii graczy,* $R(\pi_O)$ *jest optymalną*

[1] W ogólności można rozważać gry obronne Stackelberga z więcej niż dwoma graczami - wieloma obrońcami lub atakującymi, jednak w tej rozprawie uwaga skupiona będzie jedynie na wariancie z dwoma graczami, którzy w niektórych sytuacjach będą zarządzali wieloma zasobami.

(dającą mu najwyższą wypłatę) odpowiedzią Atakującego na strategię π_O, natomiast $u_g(\pi_O, \pi_A)$ oznacza wypłatę gracza $g \in \{O, A\}$ przy wyborze strategi π_O oraz π_A, odpowiednio przez Obrońcę i Atakującego.

Powyższa definicja nie jest jednak jednoznaczna, bowiem może istnieć więcej niż jedna najlepsza odpowiedź Atakującego π_A^*, a wtedy funkcja R nie będzie dobrze zdefiniowana. Dlatego też wprowadza się pojęcie Silnej Równowagi Stackelberga [10], która rozstrzyga tę sytuację.

Definicja 2. ***Silna Równowaga Stackelberga*** *to para strategii (π_O, π_A), dla których zachodzi:*

$$\begin{cases} \pi_O = \arg\max_{\pi'_O \in \Pi_O, \pi_A \in \mathbf{R}(\pi'_O)} u_O(\pi'_O, \pi_A) \\ \mathbf{R}(\pi_O) = \left\{ \pi_A \middle| (\forall \pi'_A) u_A(\pi_O, \pi_A) \geqslant u_A(\pi_O, \pi'_A) \right\} \end{cases} \tag{3.2}$$

W powyższej definicji dziedziną funkcji $\mathbf{R}$ nie jest pojedyncza strategia, a zbiór strategii. Dodatkowo w Silnej Równowadze Stackelberga w przypadku, gdy istnieje więcej niż jedna najlepsza strategia Atakującego, wybierane są te z nich, dla których wypłata Obrońcy jest najwyższa. Ta zasada może wydawać się nieintuicyjna w przypadku gier niekooperacyjnych, jakimi są gry obronne Stackelberga, ponieważ wydaje się, że gracze powinni sobie jak najbardziej "szkodzić". Alternatywą jest tzw. Słaba Równowaga Stackelberga, w której remisy strategii Atakującego rozstrzygane są w sposób odwrotny do powyższego, czyli poprzez wybór strategii, która minimalizuje wypłatę Obrońcy. Jednak udowodnione zostało, że istnieją gry, gdzie Słaba Równowaga Stackelberga nie istnieje [78]. Z tego powodu w praktyce prawie zawsze rozważana jest Silna Równowaga Stackelberga. Dodatkowym, często przytaczanym argumentem, jest również fakt, że Obrońca zazwyczaj może wybrać strategię będącą dowolnie blisko tej z Silnej Równowagi Stackelberga, która "wymusi" odpowiednią (spośród remisowych) strategię Atakującego. Z powyższych powodów, tak jak w znacznej większości prac naukowych o tej tematyce, również w niniejszej rozprawie rozważana będzie jedynie Silna Równowaga Stackelberga. W skrócie będzie ona nazywana po prostu równowagą Stackelberga lub rozwiązaniem gry.

Powyższa definicja równowagi Stackelberga posiada wiele zastosowań w sytuacjach praktycznych, szczególnie w obszarze bezpieczeństwa. Asymetria dotycząca informacji, jakie posiadają gracze, służy często do modelowania relacji między instytucjami

zapewniającymi ochronę (które pełnią rolę Obrońcy w grze), a ich potencjalnymi przeciwnikami, np. terrorystami, bandytami, złodziejami (odgrywającymi rolę Atakującego). W takich sytuacjach atakujący mają możliwość obserwowania działań obrońców, przygotowania się i wybrania dogodnego momentu i miejsca ataku. Z drugiej strony obrońcy najczęściej nie posiadają żadnej wiedzy o przeciwnikach, nie dostosowują swoich działań pod konkretny, znany im plan ataku, a jedynie mogą, dzięki wyborowi dobrej strategii obrony, minimalizować skutki potencjalnego ataku (maksymalizować oczekiwany wynik).

3.2 Gry obronne Stackelberga

Gry obronne związane są z ochroną pewnych celów. Mogą to być między innymi wartościowe przedmioty, środki transportu, budynki, pomieszczenia czy infrastruktura sieciowa. Rolą Obrońcy jest takie przydzielenie posiadanych zasobów, aby zminimalizować straty związane z ewentualnym atakiem. W rozważanych scenariuszach Obrońca nie posiada wystarczającej liczby zasobów, aby chronić wszystkie cele jednocześnie - w takim przypadku jego strategia byłaby trywialna. Musi więc dokonać wyboru, które cele i w których krokach gry mają zostać chronione. Jeśli wybrałby on strategię prostą, zgodnie z zasadą gier obronnych Stackelberga, Atakujący miałby pewność, które z celów nie są chronione i właśnie te mógłby zaatakować z gwarancją sukcesu. Dlatego Obrońca wybiera strategię mieszaną, a Atakujący nie wie, która ze składowych strategii prostych zostanie zmaterializowana, a swoją decyzję o wyborze celu może opierać jedynie na wyliczeniu oczekiwanego prawdopodobieństwa sukcesu ataku.

Lemat 1. *Dla każdej strategii mieszanej Obrońcy istnieje co najmniej jedna najlepsza (maksymalizująca jego oczekiwaną wypłatę) strategia Atakującego, która jest strategią prostą.*

Uzasadnienie prawdziwości powyższego lematu można znaleźć w publikacji [17]. Lemat ten okazuje się być bardzo istotny z praktycznego punktu widzenia w procesie poszukiwania równowagi Stackelberga. Oznacza bowiem, że nie trzeba rozważać wszystkich strategii mieszanych Atakującego, a można ograniczyć się w jego wypadku jedynie do strategii prostych, których w praktyce jest zdecydowanie mniej. Głównym

zadaniem pozostaje więc skupienie się na znalezieniu optymalnej strategii Obrońcy, a znalezienie optymalnej odpowiedzi Atakującego często sprowadza się do sprawdzenia każdej z jego możliwych strategii prostych. Rozwiązanie gry sprowadza się zatem do znalezienia odpowiedniej strategii mieszanej Obrońcy i takie podejście jest najczęściej spotykane w literaturze.

3.3 Oznaczenia

Wszystkie typy gier obronnych Stackelberga mają pewne elementy wspólne, których oznaczenia zostaną wprowadzone w tym rozdziale i będą używane w dalszej części rozprawy.

Rozważać będziemy gry m krokowe, w których udział bierze dwóch graczy: Obrońca (O) i Atakujący (A). Przez $T = \{t_1, , t_2, \ldots, t_n\}$ oznaczony będzie zbiór n celów, które są przedmiotem rozgrywki. Z każdym celem[2] $t \in T$ związane są 4 wypłaty U_t^j, $j \in \{O+, O-, A+, A-\}$, które reprezentują kolejno: nagrodę Obrońcy za schwytanie Atakującego w celu t (U_t^{O+}), karę Obrońcy z udany atak na cel t (U_t^{O-}), nagrodę Atakującego za udany atak na cel t (U_t^{A+}) oraz karę Atakującego za schwytanie go w celu t (U_t^{A-}).

Listę wszystkich możliwych strategii prostych gracza $g \in \{O, A\}$ oznaczać będziemy przez Σ_g. Wtedy strategia mieszana gracza π_g jest rozkładem prawdopodobieństwa nad Σ_g: $\pi_g = \{(\sigma_g^i, p_i)\}$, gdzie p_i jest prawdopodobieństwem zagrania strategii $\sigma_g^i \in \Sigma_g$.

Obrońca ma do dyspozycji k jednostek zasobów (w praktyce są to np. strażnicy, drony, kamery), które mogą zostać przypisane do celów. Przez $a_{js} \in T$ będziemy oznaczać cel, który został przypisany do jednostki j w kroku czasowym s ($j \in \{1, \ldots, k\}, s \in \{1, \ldots, m\}$).

Pokryciem celu (ang. *coverage*) t w kroku czasowym s (oznaczone przez $c_s(t)$) strategii mieszanej Obrońcy π_O nazywać będziemy prawdopodobieństwo następującego zdarzenia: co najmniej jedna jednostka Obrońcy jest przypisana do celu t w kroku

[2]W niektórych wariantach gier wypłaty przypisywane są graczom nie tylko w celach, lecz również w określonych okolicznościach w innych elementach gry, np. podczas spotkania graczy w tym samym wierzchołku grafu niebędącym celem.

czasowym s podczas wyboru strategii π_O: $c_s(t) = \sum_{\sigma_O^i \in \pi_O} p_i : \exists_{a_{js} \in \sigma_O^i} a_{js} = t$.

Pozostałe definicje i zasady (np. w szczególności dokładna postać strategii Obrońcy i Atakującego lub sposób liczenia wypłat graczy) zależą od konkretnego typu rozważanej gry i zostaną wprowadzone w momencie opisu poszczególnych rodzajów gier w dalszej części rozprawy.

3.4 Praktyczne zastosowania

Gry obronne Stackelberga znajdują szerokie zastosowanie praktyczne w wielu obszarach bezpieczeństwa, ochrony środowiska naturalnego, czy walki z przestępczością [55]. Szczególny wzrost zainteresowania tą tematyką obserwuje się w ciągu ostatnich 10 lat [70]. Co ważne, wiele z potencjalnych zastosowań systemów opartych o znajdowanie równowagi Stackelberga nie pozostaje jedynie w sferze teoretycznych rozważań, lecz są one faktycznie wdrażane i odgrywają ważną rolę w poprawie bezpieczeństwa kluczowej infrastruktury.

Jednym z pierwszych przykładów takiego zastosowania był system ARMOR (ang. *Assistant for Randomized Monitoring over Routes*) [60], który od 2007 roku pomaga w planowaniu kontroli bezpieczeństwa oraz układaniu harmonogramów patroli strażników w jednym z największych na świecie portów lotniczych w Los Angeles. Jako kolejny przykład można przytoczyć system PROTECT (ang. *Port Resilience Operational/Tactical Enforcement to Combat Terrorism*) [67] wykorzystywany przez Straż Wybrzeża Stanów Zjednoczonych do wyliczania optymalnego harmonogramu dla łodzi ochraniających statki w zatokach Massachusetts i Nowojorskiej. Rozwiązania oparte o gry Stackelberga były też implementowane w transporcie publicznym do optymalizacji procesu kontroli biletów. W Los Angeles podejmowanie decyzji o tym, kiedy i które linie metra powinny zostać skontrolowane wspomaga system TRUSTS (ang. *Tactical Randomization for Urban Security in Transit Systems*) [86] wdrożony w 2012 roku. Innym ważnym obszarem zastosowań gier obronnych Stackelberga, które szczególnie w ostatnich latach zyskuje na popularności, jest ochrona środowiska naturalnego, na przykład poprzez walkę z kłusownictwem [26]. Przykładem jest tu system PAWS (ang. *Protection Assistant for Wildlife Security*) [25], który organizuje codzienną pracę

strażników w Parku Narodowym Królowej Elżbiety w Ugandzie. Dzięki jego wdrożeniu udało się pięciokrotnie zwiększyć liczbę likwidowanych przez pracowników wnyków. Natomiast wprowadzenie systemu COmPASS (ang. *Conservative Online Patrol ASSistant*) [28] przyczyniło się do znacznego ograniczenia nielegalnego połowu ryb w Zatoce Meksykańskiej. Innymi obszarami zastosowań gier obronnych Stackelberga są cyberbezpieczeństwo [71, 54], walka z terroryzmem [72], ochrona granic [12], wykrywanie oszustw [52], projektowanie szczepionek [57], kontrola ruchu drogowego [66] czy testowanie oprogramowania [41].

Wymieniona wyżej lista zastosowań, mimo że obszerna, nie jest kompletna. Szerszy opis potencjalnych obszarów implementacji gier obronnych Stackelberga można znaleźć w [70].

Reasumując, podjęty w rozprawie problem ma istotne praktyczne znaczenie w ważnych aspektach życia społecznego dotyczących zapewnienia bezpieczeństwa ludziom i środowisku naturalnemu. Istnieje zatem wiele potencjalnych obszarów zastosowania proponowanego w rozprawie rozwiązania.

Rozdział 4

Przegląd istniejących rozwiązań

W niniejszym rozdziale przedstawione zostaną najczęściej występujące w literaturze podejścia do poszukiwania równowagi Stackelberga w wielokrokowych grach obronnych. W dalszej części rozprawy opisane tu metody posłużą do porównania i oceny jakości zaproponowanego przez autora algorytmu ewolucyjnego.

Podstawowym rozróżnieniem spotykanych w literaturze podejść jest podział na metody dokładne i przybliżone.

Metody dokładne wyliczają ściśle strategie wchodzące w skład równowagi Stackelberga oraz podają dokładne wartości oczekiwanych wypłat graczy. Ich działanie oparte jest najczęściej o metody programowania liniowego i całkowitoliczbowego, czyli zdefiniowanie funkcji celu oraz ograniczeń, jakie musi spełniać zwrócona odpowiedź. Do ich rozwiązywania służą najczęściej wyspecjalizowane programy komputerowe nazywane powszechnie *solverami*, które są zoptymalizowane pod kątem obliczania tak zdefiniowanych problemów[1]. Jednak takie metody są kosztowne obliczeniowo, zużywają dużo zasobów pamięciowych oraz - w praktyce - pozwalają na rozwiązanie jedynie ograniczonej klasy gier o relatywnie niewielkich rozmiarach.

Metody przybliżone zwracają rozwiązania szybciej przy mniejszym zużyciu zasobów pamięciowych i obliczeniowych, lecz otrzymywane rezultaty obarczone są pewną niedokładnością. Zwrócony wynik nie zawsze jest poszukiwaną równowagą Stackelberga, choć założeniem jest, aby był on jak najbardziej do niej zbliżony (zazwyczaj pod względem wartości oczekiwanych wypłat graczy).

[1] Więcej informacji o metodach programowania liniowego i całkowitoliczbowego, sposobach definiowania problemów i algorytmach ich rozwiązywania można znaleźć w książce [20].

Poniżej znajduje się opis kilku najbardziej istotnych podejść zaproponowanych dotychczas w literaturze. Dla przejrzystości opisu niektóre ze szczegółów prezentowanych metod zostały pominięte, można się z nimi dokładnie zapoznać w cytowanych pracach źródłowych. W opisie metod będą używane pojęcia i oznaczenia zdefiniowane w rozdziałach 2.1 i 3.3.

4.1 Metody dokładne

4.1.1 DOBSS

Pierwsza z metod dokładnych - DOBSS (ang. *Decomposed Optimal Bayesian Stackelberg Solver*) [59] oparta jest o technikę programowania liniowego i służy do poszukiwania równowagi Stackelberga w grach jednokrokowych. Dlatego też w przypadku zastosowania jej do gier wielokrokowych (o m krokach) rozważanych w rozprawie, każda możliwa m-elementowa sekwencja akcji graczy traktowana jest jako jeden ruch/akcja. DOBSS jest jedną z najbardziej znanych w literaturze metod dokładnych poszukiwania równowagi Stackelberga, która wielokrotnie służyła do porównywania jakości proponowanych rozwiązań - między innymi [33, 30, 67, 1].

Program liniowy metody DOBSS wygląda następująco:

$$\max_{q,p,x} \sum_{i\in\Sigma_O}\sum_{j\in\Sigma_A} p_{ij}u^O_{ij}, \text{ takie że} \tag{4.1}$$

$$\sum_{i\in\Sigma_O}\sum_{j\in\Sigma_A} p_{ij} = 1 \tag{4.2}$$

$$p_{ij} \in [0,1] \tag{4.3}$$

$$\sum_{j\in\Sigma_A} p_{ij} \leqslant 1 \ \forall_{i\in\Sigma_O}, \tag{4.4}$$

$$q_j \in \{0,1\}\forall_{j\in\Sigma_A}, \tag{4.5}$$

$$q_j \leqslant \sum_{i\in\Sigma_O} p_{ij} \leqslant 1, \ \forall_{j\in\Sigma_A} \tag{4.6}$$

$$\sum_{j\in\Sigma_A} q_j = 1 \tag{4.7}$$

$$0 \leqslant (x - \sum_{i \in \Sigma_O} u_{ij}^A (\sum_{k \in \Sigma_A} p_{ik})) \leqslant (1 - q_j)M, \ \forall_{j \in \Sigma_A} \tag{4.8}$$

u_{ij}^O oraz u_{ij}^A są odpowiednio wypłatą Obrońcy i Atakującego w wyniku zagrania przez nich ruchów i oraz j, Σ_O oraz Σ_A to zbiory wszystkich możliwych ruchów odpowiednio Obrońcy i Atakującego, M oznacza pewną dużą stałą (np. 10^9), q, p, x są zmiennymi programu liniowego, których wartości ulegają modyfikacji w czasie jego działania.

Główną ideą wyżej zaprezentowanego programu liniowego jest wprowadzenie zmiennych p_{ij}, które oznaczają prawdopodobieństwo zagrania ruchów i oraz j odpowiednio przez Obrońcę i Atakującego. Zmienna q_j wymusza na Atakującym zagranie dokładnie jednego ruchu, a ostatnie nierówności sprawiają, że jest on optymalny.

Mimo że metoda DOBSS gwarantuje optymalność zwracanych wyników i można ją wykorzystać do rozwiązywania gier wielokrokowych, to często w praktyce jest nieużyteczna. Wymaga ona rozważenia wszystkich ruchów, których liczba rośnie wykładniczo wraz ze wzrostem liczby kroków (każda możliwa sekwencja akcji traktowana jest jako pojedynczy ruch). Powoduje to znaczne zużycie zasobów pamięciowych i wydłuża obliczenia. Z tego względu zaproponowano inne metody dokładne, przeznaczone specjalnie do poszukiwania równowagi Stackelberga w grach wielokrokowych.

4.1.2 BC2015

Kolejna metoda dokładna również oparta jest o technikę programowania liniowego. Została ona zaproponowana w pracy [9] i będzie nazywana w tej rozprawie BC2015 (od pierwszych liter nazwisk autorów oraz roku opublikowania).

Metoda ta optymalizuje przypisanie prawdopodobieństw wszystkim możliwym sekwencjom akcji graczy, maksymalizując oczekiwaną wypłatę Obrońcy. Pełne sformułowanie ograniczeń programu liniowego wygląda następująco:

$$\max_{p,r,q,s} \sum_{z \in Z} p(z) u_O(z), \text{ takie że} \tag{4.9}$$

$$q_{I_A(\sigma_A)} = s_{\sigma_A} + \sum_{I' \in \mathcal{I}_A | \sigma_A(I') = \sigma_A} q_{I'} \quad + \sum_{\sigma_O \in \Sigma_O} r_O(\sigma_O) \bar{u}_A(\sigma_O, \sigma_A) \tag{4.10}$$

$$r_g(\emptyset) = 1 \quad \forall g \in G \tag{4.11}$$

$$r_g(\sigma_i) = \sum_{a \in A_g(I_g)} r_g(\sigma_g a) \quad \forall g \in G \; \forall I_g \in \mathcal{I}_g, \sigma_g = \sigma_g(I_g) \tag{4.12}$$

$$0 \leqslant s_{\sigma_A} \leqslant (1 - r_A(\sigma_A)) \cdot M \quad \forall \sigma_A \in \Sigma_A \tag{4.13}$$

$$0 \leqslant p(z) \leqslant r_g(\sigma_g(z)) \quad \forall g \in G \quad \forall z \in Z \tag{4.14}$$

$$1 = \sum_{z \in Z} p(z) \tag{4.15}$$

$$r_A(\sigma_A) \in \{0, 1\} \quad \forall \sigma_A \in \Sigma_A \tag{4.16}$$

$$0 \leqslant r_O(\sigma_O) \leqslant 1 \quad \forall \sigma_O \in \Sigma_O \tag{4.17}$$

Z jest zbiorem stanów terminalnych gry, $u_g(z)$ jest wypłatą gracza $g \in G$ w stanie terminalnym z, $G = \{O, A\}$ jest zbiorem graczy (Obrońca i Atakujący), $p(z)$ jest prawdopodobieństwem osiągnięcia stanu z, $\mathcal{I}_g$ jest zbiorem wszystkich zbiorów informacyjnych gracza g (I_g), $A_g(I_g)$ jest zbiorem akcji dostępnych w zbiorze informacyjnym I_g, σ_g to sekwencja akcji gracza g, Σ_g to zbiór wszystkich możliwych sekwencji akcji (strategii prostych) gracza g, $\sigma_g(v)$ oznacza sekwencję akcji gracza g, która doprowadziła do stanu v, $\bar{u}_g(\sigma_O, \sigma_A)$ to wypłata gracza g jeśli sekwencje akcji σ_O, σ_A doprowadziły do stanu terminalnego lub 0 w przeciwnym przypadku, $I_g(\sigma_g)$ jest zbiorem informacyjnym, do którego należy stan, będący wynikiem sekwencji σ_g, M oznacza pewną dużą stałą, np. 10^9, p, r, q, s są zmiennymi programu liniowego.

Zmienne $r_g(\sigma_g)$ oznaczają prawdopodobieństwo zagrania sekwencji akcji σ_g przez gracza g. W przypadku Atakującego $r_A(\sigma_A) \in \{0, 1\}$ (równanie 4.16), bo zgodnie z lematem 1 dla każdej strategii Obrońcy istnieje optymalna odpowiedź Atakującego, będąca strategią prostą. Dla Obrońcy może to być już dowolna strategia mieszana, więc $r_O(\sigma_O) \in [0, 1]$ (równanie 4.17). Równanie 4.9 określa wartość, która jest maksymalizowana, czyli jest to oczekiwana wypłata Obrońcy, wyrażona sumą po wszystkich stanach terminalnych iloczynu wypłaty w danym stanie ($u_O(z)$) i prawdopodobieństwa osiągnięcia danego stanu ($p(z)$). Równania 4.11 i 4.12 zapewniają odpowiednie przypisanie prawdopodobieństw sekwencjom poszczególnych akcji $r_g(\sigma_g)$ - prawdopodobieństwo sekwencji σ_g musi być sumą prawdopodobieństw wszystkich sekwencji, które mogą być rozszerzeniem σ_g o kolejną akcję. Równanie 4.14 łączy zmienne p i r, tj. prawdopodobieństwo osiągnięcia stanu terminalnego z ($p(z)$) nie może być większe niż

prawdopodobieństwo sekwencji prowadzącej do z. Suma prawdopodobieństw osiągnięcia poszczególnych stanów terminalnych musi być równa 1, co zapewnia równanie 4.15. Zmienne q określają wypłatę Atakującego przy zagraniu konkretnej sekwencji akcji. Równania 4.10 oraz 4.13 zapewniają, że wybrana strategia Atakującego r_A jest optymalną odpowiedzią na strategię Obrońcy.

4.1.3 C2016

Następnym algorytmem dokładnym zaproponowanym w literaturze jest C2016 [13]. Jego główna idea polega na obliczeniu tzw. skorelowanej równowagi Stackelberga, a następnie odpowiednim zmodyfikowaniu jej, w celu uzyskania docelowej równowagi Stackelberga w tradycyjnej formie. Skorelowana równowaga Stackelberga zakłada, że Obrońca (oprócz wykonywania ruchów zgodnie z przyjętą strategią) może wysyłać pewne rekomendacje widoczne dla Atakującego, a najlepszą odpowiedzią Atakującego jest podążanie za tymi sygnałami (szczegóły zostały opisane w [6, 13]). W celu policzenia skorelowanej równowagi Stackelberga C2016 używa programu liniowego zaprezentowanego poniżej.

$$\max_{p,w} \sum_{\sigma_O \in \Sigma_O} \sum_{\sigma_A \in \Sigma_A} p(\sigma_O, \sigma_A) \bar{u}_O(\sigma_O, \sigma_A), \text{ takie że} \tag{4.18}$$

$$p(\emptyset, \emptyset) = 1 \quad 0 \leqslant p(\sigma_O, \sigma_A) \leqslant 1 \tag{4.19}$$

$$p(\sigma_O(I), \sigma_A) = \sum_{a \in A(I)} p(\sigma_O(I)a, \sigma_A) \quad \forall I \in \mathcal{I}_O, \forall \sigma_A \in rel(\sigma_O(I)) \tag{4.20}$$

$$p(\sigma_O, \sigma_A(I)) = \sum_{a \in A(I)} p(\sigma_O, \sigma_A(I)a) \quad \forall I \in \mathcal{I}_A, \forall \sigma_O \in rel(\sigma_A(I)) \tag{4.21}$$

$$w(\sigma_A) = \sum_{\sigma_O \in rel(\sigma_A)} p(\sigma_O, \sigma_A) \bar{u}_A(\sigma_O, \sigma_A) + \sum_{I \in \mathcal{I} | \sigma_A(I) = \sigma_A} \sum_{a \in A_A(I)} w(\sigma_A a) \tag{4.22}$$

$$w(I, \sigma_A) \geqslant \sum_{\sigma_O \in rel(\sigma_A)} p(\sigma_O, \sigma_A) \bar{u}_A(\sigma_O, \sigma_A(I)a) + \sum_{I' \in \mathcal{I}_A | \sigma_A(I') = \sigma_A(I)a} w(I', \sigma_A),$$
$$\forall I \in \mathcal{I}_A, \forall \sigma_A \in \bigcup_{h \in I} rel(\sigma_O(h)), \forall a \in A(I) \tag{4.23}$$

$$w(\sigma_A(I)a) = w(I, \sigma_A(I)a) \quad \forall I \in \mathcal{I}_A, \forall a \in A(I) \tag{4.24}$$

Przez $rel(\sigma_g)$ oznaczony jest zbiór wszystkich sekwencji przeciwnika gracza g, które tworzą z σ_g parę sekwencji istotnych. Para sekwencji (σ_O, σ_A) jest *istotna* (ang. *relevant*), jeśli $\sigma_A = \emptyset$ lub $\exists_{v,v' \in D_v, v' \sqsubseteq v}(\sigma_O = \sigma_O(v) \wedge v' \in I_A(\sigma_A))$, gdzie D_v jest zbiorem wszystkich wierzchołków drzewa gry (patrz definicja gry w formie ekstensywnej z rozdziału 2.1), a oznaczenie $v' \sqsubseteq v$ oznacza, że wierzchołek v' leży na ścieżce od korzenia drzewa gry do wierzchołka v (szczegółowy opis i uzasadnienie sekwencji istotnych można znaleźć w [13]). Równanie 4.18 reprezentuje oczekiwaną wypłatę Obrońcy, która jest maksymalizowana. Równania 4.19-4.21 mają na celu zagwarantowanie poprawności przyjętej strategii, tzn. $p(\sigma_O, \sigma_A)$, czyli prawdopodobieństwo zagrania danej sekwencji, jest sumą prawdopodobieństw zagrania sekwencji ją rozszerzających. w są zmiennymi, które gwarantują optymalność odpowiedzi Atakującego (równania 4.22 i 4.23). $w(\sigma_A)$ oznacza wypłatę Atakującego w przypadku wyboru strategii σ_A i podążaniu za rekomendacjami Obrońcy (związanej ze skorelowaną równowagą Stackelberga).

Strategia wyliczona przy pomocy powyższego programu liniowego jest równowagą skorelowaną, tzn. określa prawdopodobieństwa pary sekwencji - $p(\sigma_O, \sigma_A)$. Aby wyliczyć klasyczną równowagę Stackelberga dodatkowa procedura (patrz Algorytm 1 w [13]) iteracyjnie dodaje kolejne ograniczenia, które zamieniają prawdopodobieństwo zagrania niektórych z sekwencji na 0 lub 1. Program liniowy jest modyfikowany, aby zmniejszyć liczbę rekomendacji w wyjściowej skorelowanej równowadze Stackelberga i doprowadzić ją do docelowej (nieskorelowanej) równowagi Stackelberga. Podejście to, podobnie jak metoda BC2015, działa w czasie wykładniczym, lecz w przypadku niektórych gier wykazuje lepsze własności obliczeniowe (krótszy czas obliczeń i mniejsze zużycie pamięci).

4.2 Metody przybliżone

4.2.1 O2UCT

Podstawą działania pierwsza z omawianych metod przybliżonych jest algorytm *Monte Carlo Tree Search* [11] (MCTS). W dalszej części rozprawy to podejście będzie nazywane O2UCT (od *double-oracle UCT sampling*). Zostało ono zaproponowane w pracach [34, 35].

Technika MCTS skupia się na analizie najbardziej obiecujących ruchów poprzez tworzenie poddrzewa gry za pomocą losowego próbkowania przestrzeni przeszukiwań. Jej działanie można podzielić na 4 fazy: wybór, ekspansja, symulacja oraz propagacja. W pierwszej fazie wybierana jest losowa ścieżka w zbudowanym dotychczas drzewie zgodnie z przyjętą strategią, która ma na celu zachowanie równowagi pomiędzy eksploatacją obiecujących ruchów (o wysokiej średniej wypłacie) a eksploracją ruchów, które dotychczas były sprawdzone w niewielkim stopniu. Aby zachować odpowiednią proporcję między tymi podejściami zaproponowano technikę wyboru kolejnych wierzchołków nazywaną *Upper Confidence Bounds applied to Trees* (UCT) [39]. Zgodnie z nią selekcja ścieżki w drzewie następuje na podstawie dwóch wartości: średniej wypłaty związanej z zagraniem ruchu przypisanego do i-tej gałęzi drzewa (Q_i) oraz licznika odwiedzin tej gałęzi (n_i). Na kolejnych poziomach drzewa wybierana jest ta gałąź, dla której wartość wyrażenia $Q_i + \frac{c\sqrt{\log \sum_i n_i}}{n_i}$ jest najwyższa. W momencie dotarcia do liścia drzewa, następuje faza ekspansji, która dodaje do budowanego drzewa gry węzeł będący losowym następnikiem osiągniętego liścia. Kolejnym krokiem jest symulacja, która wybiera losowo kolejne gałęzie z drzewa gry, aż do momentu osiągnięcia stanu terminalnego. Dzięki temu uzyskiwana jest ocena wybranej ścieżki i przeprowadzana aktualizacja wartości Q_i wzdłuż niej do korzenia drzewa.

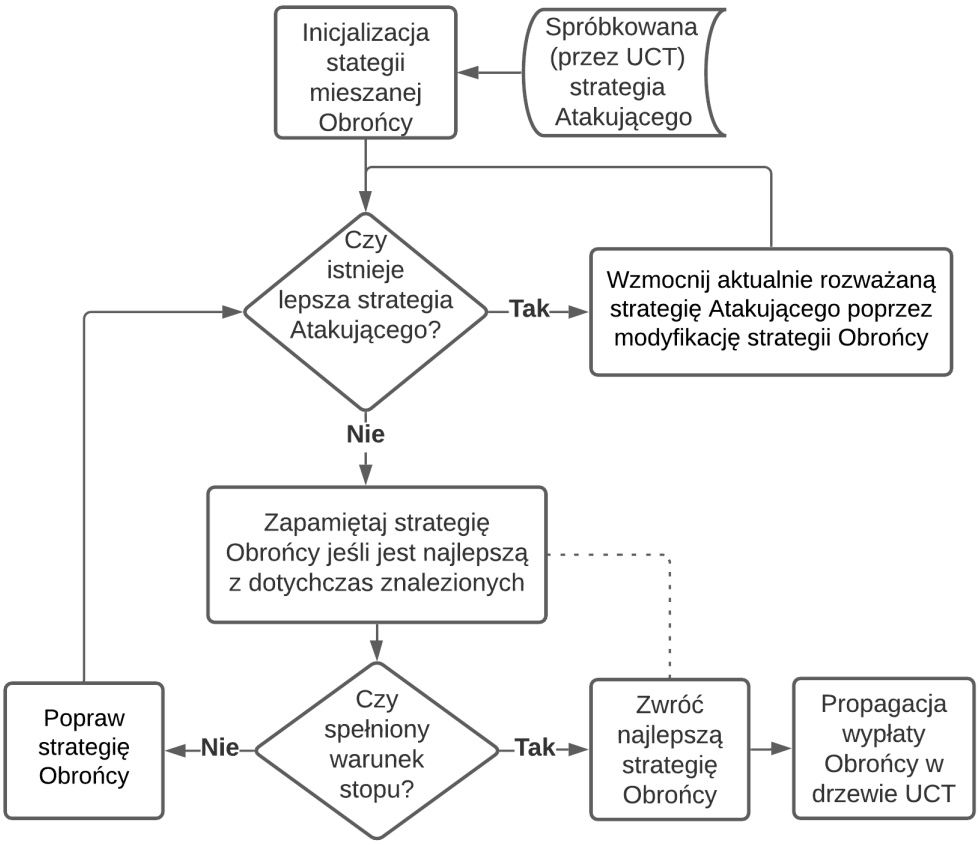
Algorytm O2UCT działa zgodnie z wyżej opisanymi zasadami. Bazuje na ukierunkowanym próbkowaniu przestrzeni strategii Atakującego (przy użyciu metody UCT) z poszukiwaniem najlepszej strategii Obrońcy, dla której dana (spróbkowana) strategia Atakującego będzie optymalną odpowiedzią. Metoda buduje drzewo strategii Atakującego. Kolejne jego poziomy reprezentują możliwe do wyboru akcje w kolejnych krokach czasowych.

Algorytm O2UCT można podzielić na 2 poziomy: zewnętrzny i wewnętrzny. Na poziomie zewnętrznym wybierana jest strategia Atakującego przy wykorzystaniu wyżej opisanej techniki UCT. Ta strategia oceniania jest w poziomie wewnętrznym algorytmu O2UCT. Oceną jest wypłata Obrońcy, która jest następnie propagowana w tworzonym drzewie w kierunku korzenia i wpływa na wybór ścieżek w kolejnych iteracjach algorytmu.

Poziom wewnętrzny algorytmu O2UCT rozwiązuje problem optymalizacji z ogra-

niczeniami. Każda spróbkowana strategia jest oceniana poprzez wygenerowanie jak najlepszej (dającej jak najwyższą wypłatę) strategii mieszanej Obrońcy, która musi spełniać następujący warunek: najlepszą odpowiedzią Atakującego na tę strategię jest oceniana strategia Atakującego. Jeśli ten warunek nie jest zachowany, to strategia Obrońcy jest odpowiednio modyfikowana, aby go spełnić. W przeciwnym przypadku strategia Obrońcy jest poprawiana, aby zwiększyć jego oczekiwaną wypłatę. Procedura modyfikowania i poprawiania są wykonywane wielokrotnie aż do spełnienia warunku stopu.

Rysunek 4.1 przedstawia ogólny schemat działania procedury poszukiwania najlepszej strategii Obrońcy dla zadanej strategii Atakującego.



Rysunek 4.1: Schemat procedury poszukiwania najlepszej strategii Obrońcy dla zadanej strategii Atakującego w algorytmie O2UCT.

Eksperymenty przeprowadzone w [34, 35] pokazały, że O2UCT zwraca dobre jakościowo rozwiązania (bliskie optymalnym) w czasie krótszym niż metody dokładne.

4.2.2 CBK2017

Główna zasada drugiej metody przybliżonej - CBK2017 [14] - polega na iteracyjnym rozbudowywaniu drzewa gry i liczeniu wyników dla zredukowanych, mniejszych gier, zamiast od razu rozważać wszystkie dostępne możliwości. Metoda ta rozwiązuje gry w formie ekstensywnej (patrz rozdział 2.1). Pseudokod metody wraz z pomocniczymi procedurami zaprezentowany został jako Algorytmy 4.1, 4.2 i 4.3.

Algorytm 4.1: Pseudokod metody CBK2017.

```
1  CBK2017 (G)
2      (M, Σ') ← ExpandGadget(root(G, ∅)
3      expansionNeeded ← |M| > 0
4      while expansionNeeded do
5          for state h ∈ M do
6              Σ' ← Σ' ∪ CreateGadget(h)
7          (value, p) ← CBK2017(Σ')
8          M ← ∅
9          E ← getGadgetsToExpand(p)
10         expansionNeeded ← False
11         for state h ∈ E do
12             (M', Σ') ← ExpandGadget(h, Σ')
13             M ← M ∪ M'
14             if |M'| > 0 then
15                 expansionNeeded ← True
16     return (value, p)
```

Algorytm 4.2: Tworzenie obiektu *gadget* w metodzie CBK2017.

```
1  CreateGadget (state h)
2      Σ̄ ← ∅
3      Z_h ← getLeafsUnder(h)
4      Z_h ← getUpperConvexHull(Z_h)
5      for state z ∈ Z_h do
6          a_{z,h} ← createNewAction(z, h)
7          Σ̄ ← Σ̄ ∪ seq_O(h)a_{z,h}
8      return Σ̄
```

Drzewo gry jest redukowane poprzez zastosowanie abstrakcyjnych struktur nazywanych *gadget*. Reprezentują one poddrzewa oryginalnego drzewa gry i składają się z

Algorytm 4.3: Rozszerzanie obiektu *gadget* w metodzie CBK2017.

ExpandGadget (h, Σ')
  $\Sigma' \leftarrow \Sigma' \setminus \{\sigma \in \Sigma' : \exists_z \sigma = \mathrm{seq}_O(h) a_{z,h}\}$
  $H_h \leftarrow$ getShallowestLeaderStates(h)
  for *state* $g \in H_h$ **do**
    $\bar{\Sigma}_O \leftarrow \{\sigma \in \Sigma_O : \sigma \sqsubseteq \sigma_O(g)\}$
    $\bar{\Sigma}_A \leftarrow \{\sigma \in \Sigma_A : \sigma \sqsubseteq \sigma_A(g)\}$
    $\Sigma' \leftarrow \Sigma' \cup \bar{\Sigma}_O \cup \bar{\Sigma}_A$
  return (H_h, Σ')

wierzchołka, w którym akcję wykonuje Obrońca oraz listy akcji, które prowadzą bezpośrednio do stanu terminalnego gry, będącego reprezentacją podzbioru stanów terminalnych w oryginalnym (niezredukowanym) drzewie. Następnie każda z tak okrojonych gier rozwiązywana jest algorytmem C2016 opisywanym we wcześniejszej części tego rozdziału. W wyniku dzielenia i łączenia drzewa gry część strategii może zostać pominięta w rozważaniach, dlatego też CBK2017 jest metodą przybliżoną, która nie daje gwarancji otrzymania optymalnego rezultatu.

Szczegółowy opis metody CBK2017 oraz przedstawionych algorytmów można znaleźć w [14].

4.3 Podsumowanie

Przedstawione w tym rozdziale metody poszukiwania równowagi w wielokrokowych grach obronnych Stackelberga nie są jedynymi, które zostały dotychczas opisane w literaturze. Jednak cechą, która wyróżnia wymienione tutaj podejścia, jest uniwersalność, pozwalająca na ich wykorzystanie do gier o różnej charakterystyce i zasadach. Istnieją inne metody przybliżonego rozwiązywania gier obronnych Stackelberga, lecz ich zastosowanie ograniczone jest do pewnej wąskiej klasy gier o określonych regułach. Wykorzystanie tych metod do pozostałych rodzajów gier wymaga istotnych modyfikacji lub jest niemożliwe. Działanie takich metod zazwyczaj opiera się o konkretne zasady gry i jest zoptymalizowane pod tym kątem. Zrozumienie tych podejść wymaga przytoczenia reguł rozwiązywanej gry. Dlatego też opis takich metod przytoczony będzie w dalszej części rozprawy w rozdziale 6. Przedstawia on wyniki eksperymentalne dla

różnych wariantów gier obronnych Stackelberga i porównanie wyników proponowanego w rozprawie algorytmu ewolucyjnego z konkurencyjnymi podejściami.

Większość z metod poszukiwania równowagi Stackelberga w grach obronnych oparta jest o programowanie liniowe i całkowitoliczbowe z ograniczeniami. Podejścia dokładne najczęściej wymagają przejrzenia wszystkich możliwych strategii graczy i przechowywania całej reprezentacji drzewa gry w pamięci, co w niektórych przypadkach jest niemożliwe ze względu na ograniczenia czasowe lub sprzętowe. Metody przybliżone adresują ten problem poprzez rozważanie tylko wybranych poddrzew i ograniczenie obliczeń jedynie do pewnego podzbioru strategii graczy. Sposób wyboru tego podzbioru jest kluczowym elementem tych metod i determinuje jakość otrzymywanych rozwiązań oraz czas działania.

Rozdział 5

Algorytmy ewolucyjne w Grach Obronnych Stackelberga

5.1 Motywacja

Znalezienie równowagi Stackelberga jest problemem NP-trudnym [17], nie jest znany więc żaden algorytm, który rozwiązywałby ten problem efektywnie (w czasie wielomianowym). Znalezienie równowagi Stackelberga (optymalnych strategii Obrońcy i Atakującego) wymaga dużych nakładów obliczeniowych. Metody dokładne opisane w poprzednim rozdziale nie są zdolne do wyliczenia optymalnego rozwiązania w akceptowalnym czasie dla gier o większych rozmiarach, które często występują w praktycznych zastosowaniach. W wielu takich przypadkach pewne odstępstwo od optymalnego rezultatu jest dopuszczalne - ważniejsze jest znalezienie dobrego rozwiązania w określonym (niedługim) czasie niż poświęcenie dużych zasobów obliczeniowych na znalezienie rozwiązania trochę lepszego. Z tego powodu, w rzeczywistych scenariuszach można zastosować algorytmy aproksymacyjne, które potrafią "szybko" zwrócić strategie dobrej jakości, bliskie optymalnym. Prezentowane w tej rozprawie podejście również wpisuje się w ten scenariusz - zaproponowany zostanie efektywny algorytm przybliżony, który pozwala na policzenie znacznie większych instancji problemów niż metody dokładne.

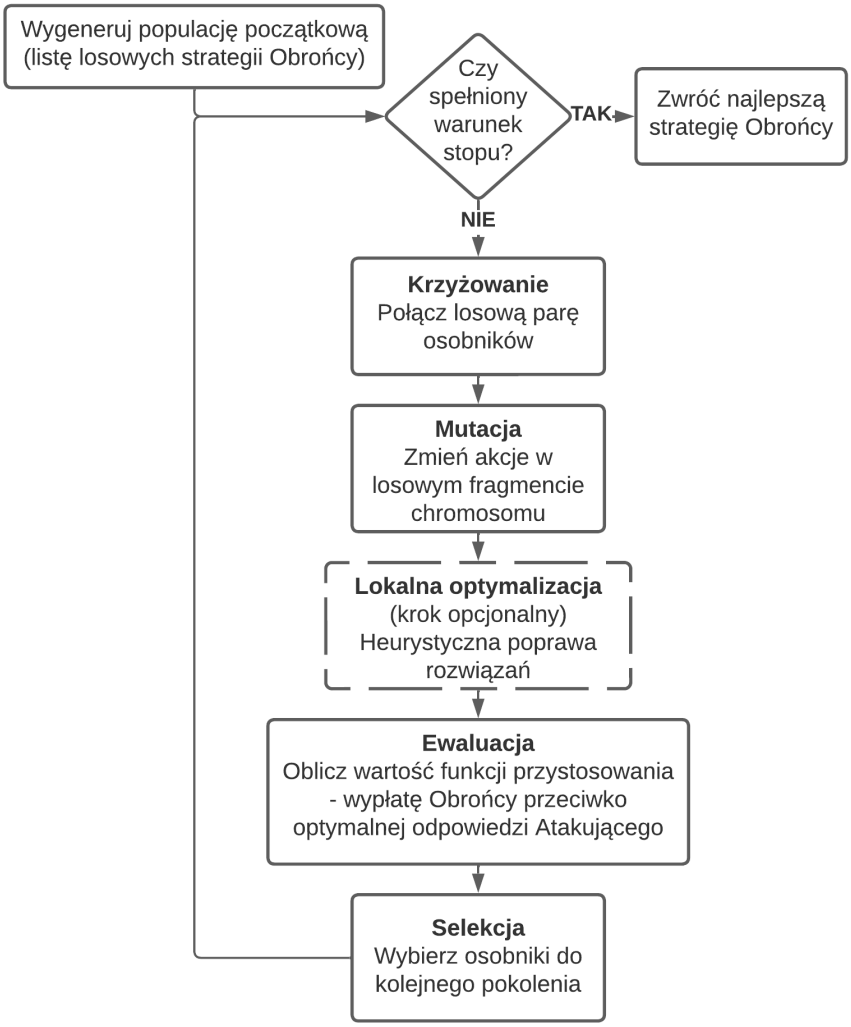
Poszukiwanie równowagi Stackelberga jest dwupoziomowym problemem optymalizacyjnym, w którym optymalizujemy strategię Obrońcy (problem zewnętrzny) przy założeniu optymalnej strategii Atakującego (problem wewnętrzny). Problem wewnętrzny jest stosunkowo prosty do rozwiązania, ponieważ zawsze istnieje co najmniej jedna

strategia **prosta** Atakującego, która jest optymalną odpowiedzią na strategię Obrońcy (patrz Lemat 1). W najprostszym podejściu wystarczy więc sprawdzić wszystkie strategie proste Atakującego i wybrać najlepszą z nich. Dużo bardziej skomplikowanym zadaniem jest znalezienie optymalnej strategii Obrońcy. Tutaj wybór następuje spośród wszystkich możliwych strategii mieszanych, których jest nieskończenie wiele. W dodatku przestrzeń przeszukiwań jest ciągła między innymi ze względu na konieczność ustalenia optymalnych prawdopodobieństw poszczególnych strategii prostych. Jedną z popularnych w literaturze i skutecznych w praktyce technik poszukiwania optimum w takiej przestrzeni są algorytmy ewolucyjne. Z tego powodu to właśnie ta metoda została wybrana jako obiecujące i potencjalnie skuteczne narzędzie znajdowania równowagi w grach obronnych Stackelberga. Wedle najlepszej wiedzy autora, zaprezentowane w tej rozprawie podejście jest pierwszym zastosowaniem algorytmów ewolucyjnych do rozwiązywania tego problemu.

5.2 Ogólny schemat

W dalszej części rozdziału zostanie zaprezentowana ogólna postać algorytmu ewolucyjnego rozwiązującego gry obronne Stackelberga, który nazywany będzie *EASG* jako skrót od angielskiej nazwy *Evolutionary Algorithm for Security Games*. Jest to możliwie uniwersalna postać zaproponowanego podejścia, która może być łatwo zmodyfikowana i dopasowana do różnych typów gier, co zostanie przedstawione w kolejnym rozdziale.

Generalny schemat proponowanego systemu jest zgodny z typową budową algorytmu ewolucyjnego przedstawionego w rozdziale 2.3.7. Na wstępie tworzona jest populacja początkowa o ustalonym rozmiarze. Następnie, do momentu spełnienia warunku stopu, generowane są kolejne pokolenia populacji poprzez zastosowanie operatorów ewolucyjnych: mutacji, krzyżowania i selekcji. Ogólny schemat proponowanego algorytmu został zaprezentowany na rysunku 5.1, a szczegóły działania poszczególnych operacji są opisane w kolejnych rozdziałach.



Rysunek 5.1: Ogólny schemat proponowanego rozwiązania.

5.3 Kodowanie rozwiązań

Każdy chromosom (osobnik) reprezentuje pewną strategię mieszaną Obrońcy, która jest potencjalnym rozwiązaniem problemu - wraz z optymalną odpowiedzią Atakującego wchodzi w skład równowagi Stackelberga. Zakodowane rozwiązania mają postać wektorów złożonych ze strategii prostych σ_i^q oraz odpowiadających im prawdopodobieństw p_i^q. Chromosom CH_q można wyrazić wzorem:

$$CH_q = \{(\sigma_1^q, p_1^q), (\sigma_2^q, p_2^q), \ldots, (\sigma_{l_q}^q, p_{l_q}^q)\}, \quad \sum_{i=1}^{l_q} p_i^q = 1, \tag{5.1}$$

gdzie q jest indeksem/identyfikatorem chromosomu w populacji, l_q oznacza długość chromosomu CH_q, czyli liczbę strategii prostych wchodzących w skład strategii mieszanej reprezentowanej przez chromosom.

Konkretna postać strategii prostej σ_i^q zależy od definicji konkretnego typu gry. Najczęściej jest to lista akcji Obrońcy w kolejnych krokach czasowych: $\sigma_i^q = (a_1, a_2, \ldots, a_m)$ (m oznacza liczbę (limit) kroków czasowych podaną w definicji gry).

5.4 Populacja początkowa

Chromosomy w populacji początkowej zawierają jedynie pojedyncze strategie proste, tzn. $\forall_q\, l_q = 1 \wedge p_1^q = 1$. Te strategie generowane są w sposób losowy według następującej procedury. W każdym kolejnym kroku czasowym (począwszy od pierwszego) wybierana jest losowa (z rozkładem jednostajnym) akcja spośród wszystkich możliwych. Ten proces wykonywany jest niezależnie dla każdego nowego chromosomu, dopóki populacja (liczba chromosomów) nie osiągnie docelowego rozmiaru (N). Algorytm 5.1 przedstawia pseudokod procedury generującej populację początkową.

Algorytm 5.1: Generowanie populacji początkowej.

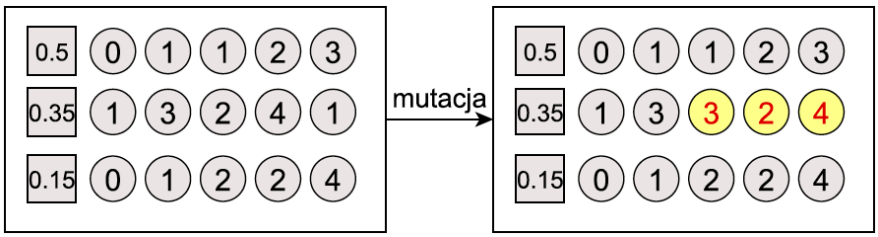
```
1  GenerateInitialPopulation ()
2      P ← ∅
3      while |P| < N do
4          σ ← ()
5          for i ∈ [1, m] do
6              a_i ← rand(A_O(σ)) // losowa akcja po wykonaniu sekwencji σ
7              σ ← σ a_i          // rozszerzenie sekwencji o akcję a_i
8          CH = {(σ, 1)}
9          P ← P ∪ {CH}
10     return P
```

5.5 Mutacja

Mutacja polega na losowym zaburzeniu wybranego podzbioru chromosomów. Operator ten stosowany jest do każdego chromosomu niezależnie, z prawdopodobieństwem p_m. Innymi słowy, każdy z osobników w populacji będzie z prawdopodobieństwem p_m

poddany mutacji.

Mutacja na początku wybiera losową strategię prostą σ_i^q spośród wszystkich strategii wchodzących w skład chromosomu. Następnie wybierany jest losowy krok czasowy s i począwszy od akcji a_s aż do ostatniej akcji a_m każda kolejna akcja zastępowana jest losową akcją spośród wszystkich dostępnych akcji w danym stanie gry. Analogiczna procedura była używana podczas tworzenia nowych chromosomów do populacji początkowej. Wynikiem mutacji strategii σ_i^q jest strategia $\sigma'^q_i = (a_1, a_2, \ldots, a_{s-1}, a'_s, a'_{s+1}, \ldots, a'_m)$. Przykładowa mutacja zaprezentowana jest na rysunku 5.2.



Rysunek 5.2: Przykład działania mutacji. Akcje oznaczone są liczbami naturalnymi w kołach, liczby w kwadratach reprezentują prawdopodobieństwa poszczególnych strategii prostych. Mutacji poddawana jest druga strategia prosta począwszy od kroku czasowego $s = 3$.

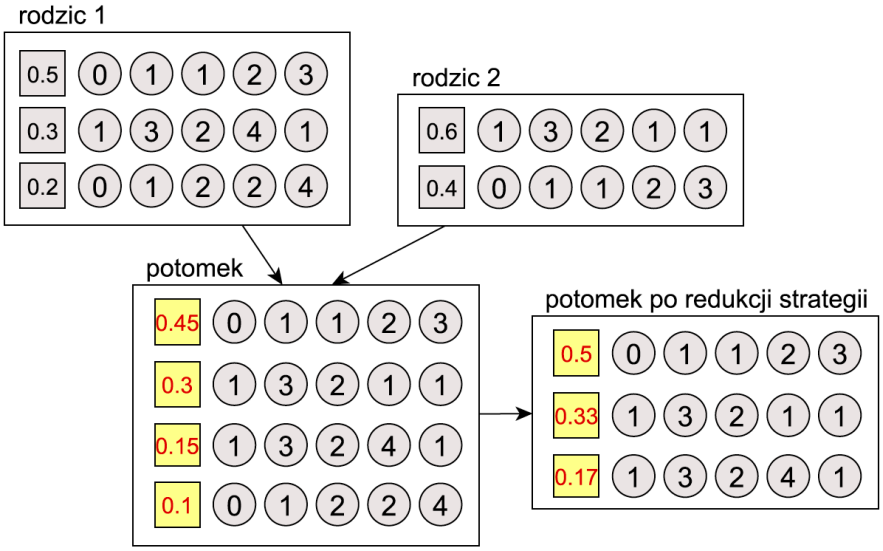
Rolą mutacji jest odkrywanie nowych, potencjalnie lepszych rozwiązań poprzez szersze przeszukiwania przestrzeni rozwiązań.

5.6 Krzyżowanie

Krzyżowanie łączy osobniki w pary i wymienia informacje między nimi. Na początku wybierany jest losowy podzbiór $\lfloor p_k N \rfloor$ osobników z aktualnej populacji. p_k jest parametrem algorytmu oznaczającym współczynnik krzyżowania. Jest on analogią do parametru prawdopodobieństwa mutacji p_m i w praktyce oznacza prawdopodobieństwo, z jakim dany chromosom będzie brał udział w procesie krzyżowania. Wybrane wcześniej $\lfloor p_k N \rfloor$ osobników dobieranych jest losowo w pary. W przypadku nieparzystej liczby losowo wybrany osobnik jest odrzucany i nie bierze udziału w tym procesie. W dalszej kolejności, z każdej pary tworzony jest jeden nowy osobnik "potomny" według następującej procedury. Wszystkie strategie proste z chromosomów obu "rodziców" łączone są w jedną strategię mieszaną, a odpowiadające im prawdopodobieństwa są

zmniejszane o połowę. Wynikiem krzyżowania chromosomów CH_1 i CH_2 jest chromosom $CH_{1-2} = \{(\sigma_1^1, \frac{p_1^1}{2}), \ldots, (\sigma_{l_1}^1, \frac{p_{l_1}^1}{2}), (\sigma_1^2, \frac{p_1^2}{2}), \ldots, (\sigma_{l_2}^2, \frac{p_{l_2}^2}{2})\}$.[1]

Wielokrotne powtarzanie takiej procedury (w kolejnych pokoleniach) doprowadziłoby do powstania strategii mieszanych zawierających bardzo dużą liczbę strategii prostych z małymi prawdopodobieństwami, co jest zjawiskiem niepożądanym. Dlatego też każda strategia prosta σ_i^q z nowego chromosomu-potomka (z wyjątkiem strategii z przypisanym najwyższym prawdopodobieństwem) jest usuwana z prawdopodobieństwem równym $(1 - p_i^q)^2$. Oznacza to, że im dana strategia prosta ma niższe prawdopodobieństwo tym większa jest szansa na jej usunięcie. Na końcu prawdopodobieństwa pozostałych nieusuniętych strategii prostych są normalizowane, aby ich suma wynosiła 1. Na rysunku 5.3 przedstawiony jest przykład operacji krzyżowania.



Rysunek 5.3: Przykład działania krzyżowania. Akcje oznaczone są liczbami naturalnymi w kołach, liczby w kwadratach reprezentują prawdopodobieństwa poszczególnych strategii prostych. W kroku redukcji usunięto ostatnią strategię, a prawdopodobieństwa pozostałych zostały znormalizowane.

Krzyżowanie wprowadza rodzaj eksploatacji przestrzeni przeszukiwań - wygenerowana nowa strategia mieszana jest kombinacją znalezionych wcześniej rozwiązań.

[1]Jeżeli w chromosomach CH_1 i CH_2 istnieją dwie identyczne strategie proste, to w wynikowym chromosomie znajduje się tylko jedna ich instancja z zsumowanym prawdopodobieństwem.

5.7 Lokalna optymalizacja

Procedura lokalnej optymalizacji ma na celu zachłanną poprawę rozwiązań wygenerowanych w wyniku zastosowania operatorów mutacji i krzyżowania. Operatory te działają w sposób losowy, więc szansa wygenerowania przez nie rozwiązania będącego optimum lokalnym jest mała. Założeniem lokalnej optymalizacji jest modyfikacja tych rozwiązań i "przesunięcie" ich w kierunku optimum lokalnego. Są to zazwyczaj różne rodzaje heurystyk. Konkretna realizacja zależy od typu rozwiązywanej gry, jej definicji i zasad. Dlatego też w tym miejscu (gdzie opisywany jest uniwersalny schemat proponowanego algorytmu) nie jest wskazywana żadna sprecyzowana postać procedury lokalnej optymalizacji. Będą one omówione w dalszych rozdziałach podczas opisu adaptacji prezentowanego schematu do poszczególnych typów gier. Zastosowanie lokalnej optymalizacji jest krokiem opcjonalnym, który nie zawsze musi być wdrożony.

5.8 Ewaluacja

Wartością funkcji przystosowania osobnika jest oczekiwana wypłata Obrońcy w przypadku zagrania strategii mieszanej zakodowanej w chromosomie. Zgodnie z Lematem 1 zawsze istnieje przynajmniej jedna strategia prosta Atakującego, która jest optymalną odpowiedzią Atakującego. Zatem, aby ocenić daną strategię mieszaną Obrońcy π_O (wyliczyć wypłatę Obrońcy) wystarczy przeszukać zbiór wszystkich możliwych strategii prostych atakującego Σ_A i dla każdej z nich (σ_A) wyliczyć wypłatę Atakującego $u_A(\pi_O, \sigma_A)$, a następnie wybrać tę strategię Atakującego, dla której ta wypłata jest najwyższa (rozstrzygając remisy na korzyść Obrońcy zgodnie z definicją silnej równowagi Stackelberga przytoczoną w rozdziale 3.1). Znaleziona w ten sposób najlepsza odpowiedź Atakującego na strategię π_O ($R(\pi_O) = \sigma_{bestA}$) służy ostatecznie do wyliczenia oczekiwanej wypłaty Obrońcy: $u_O(\pi_O, \sigma_{bestA})$, która stanowi wartość funkcji przystosowania. Algorytm 5.2 prezentuje pseudokod procedury ewaluacji strategii Obrońcy zakodowanej w chromosomie.

Często w praktyce liczba strategii prostych Atakującego jest nieduża i koszt obliczeniowy sprawdzenia wszystkich z nich jest akceptowalny. Jednak w rozprawie rozważane są również scenariusze, gdy ten warunek nie jest spełniony i przejrzenie wszystkich stra-

Algorytm 5.2: Ewaluacja strategii Obrońcy π_O.

```
1 EvaluateSolution (π_O)
2     σ_bestA ← null
3     for σ_A ∈ Σ_A do
4         if σ_bestA = null or u_A(π_O, σ_A) > u_A(π_O, σ_bestA) or
5         (u_A(π_O, σ_A) = u_A(π_O, σ_bestA) and u_O(π_O, σ_A) > u_O(π_O, σ_bestA)) then
6             σ_bestA ← σ_A
7     return u_O(π_O, σ_bestA)
```

tegii prostych Atakującego jest zbyt czasochłonne lub praktycznie niemożliwe. Jedną z metod rozwiązania takiej sytuacji jest użycie przybliżonej metody oceny jakości rozwiązania - oszacowania wypłaty Obrońcy. W dalszej części rozprawy zaproponowane zostały dwa takie podejścia:

(1) wykorzystanie sztucznej sieci neuronowej uczonej estymowania wypłaty Obrońcy na podstawie jego strategii oraz definicji gry - rozdział 8.4,

(2) zastosowanie podejścia koewolucyjnego, w którym populacja strategii Obrońcy jest oceniana względem wybranych, reprezentatywnych strategii z równolegle rozwijanej populacji strategii Atakującego - rozdział 9.

5.9 Selekcja

Rolą selekcji jest promocja lepiej przystosowanych osobników do kolejnego pokolenia. Na wstępie e osobników (nazywanych *elitą*) z najwyższą wartością funkcji przystosowania (wypłaty Obrońcy) spośród wszystkich osobników z aktualnej populacji (włączając również te osobniki, które powstały w wyniku mutacji i krzyżowania) jest bezwarunkowo przenoszona do kolejnego pokolenia - nowej populacji. W ten sposób zyskujemy pewność, że najlepsze znalezione rozwiązania nie będą zapomniane w procesie ewolucji oraz będą brały udział w reprodukcji w kolejnych pokoleniach.

Następnie procedura selekcji przeprowadza szereg *turniejów binarnych*, w wyniku których uzupełniana jest populacja nowego pokolenia do momentu osiągnięcia zakładanej liczby osobników - N. Każdy turniej polega na wylosowaniu z aktualnej populacji dwóch osobników - uczestników turnieju. Następnie z prawdopodobieństwem p_s osobnik z wyższą wartością funkcji przystosowania jest kopiowany do nowego pokolenia. W

przeciwnym przypadku promowany jest drugi z osobników (z niższą wartością funkcji przystosowania). p_s jest parametrem algorytmu nazywanym presją selekcji. Niezależnie od wyniku turnieju oba osobniki mogą uczestniczyć w kolejnych turniejach, tzn. są zwracane do początkowej puli, z których losowane są pary turniejowe. Oznacza to, że możliwa jest wielokrotna promocja tego samego osobnika - w nowym pokoleniu są tworzone jego kopie. Schemat procedury selekcji turniejowej został zaprezentowany w rozdziale 2 jako algorytm 2.1.

5.10 Warunek stopu

Działanie algorytmu (generowanie kolejnych pokoleń) jest przerywane w momencie spełnienia przynajmniej jednej z następujących przesłanek:
- brak poprawy najlepszego znalezionego rozwiązania przez g_k kolejnych pokoleń,
- osiągnięcie limitu g_l wygenerowanych pokoleń.

Wartości parametrów g_k i g_l ustalane są eksperymentalnie na podstawie obserwacji rozwoju populacji w procesie wstępnej parametryzacji algorytmu.

Rozdział 6

Eksperymenty

W ostatnich latach zaproponowano wiele rodzajów gier obronnych Stackelberga (GOS) różniących się zasadami. GOS możemy podzielić na jednokrokowe lub wielokrokowe, z pełną lub niepełną informacją, z pełną lub częściową obserwowalnością, z dyskretną lub ciągłą przestrzenią wyboru strategii, deterministyczne lub niedeterministyczne, z pamięcią wyniku poprzednich rozgrywek lub bez niej, z pełną lub niepełną racjonalnością. Mnogość wariantów uniemożliwia rzetelne przetestowanie zaproponowanej metody na wszystkich z nich. Dlatego też w rozprawie do testów zostało wybranych 5 typów gier, które są często używane w literaturze lub mają pewną specyficzną charakterystykę uwydatniającą jakiś aspekt proponowanej metody.

Poniżej zaprezentowane zostaną eksperymenty weryfikujące skuteczność przedstawionego w poprzednim rozdziale algorytmu ewolucyjnego (EASG) dla 5 różnych typów gier obronnych Stackelberga. Każdy rodzaj gry zostanie najpierw opisany pod względem jej zasad, następnie omówione zostaną wymagane modyfikacje metody EASG adaptujące ją do danego rodzaju gry, a finalnie zademonstrowane będą wyniki metody wraz z komentarzem.

Dla wszystkich eksperymentów przyjęto stałe wartości parametrów algorytmu zaprezentowane w tabeli 6.1. Sposób wyznaczenia tych parametrów, analiza ich wpływu na działanie algorytmu oraz dyskusja ich znaczenia zaprezentowane zostaną w rozdziale 7.1. W przypadku metod niedeterministycznych wszystkie prezentowane wyniki zostały uśrednione z 30 niezależnych uruchomień algorytmu dla każdego przypadku testowego. W przypadku metod przybliżonych założono, że dla danej instancji proble-

mu osiągnięty został wynik optymalny, jeśli różnica pomiędzy wynikiem optymalnym a zwróconą wartością (średnią z 30 uruchomień) była mniejsza niż $\varepsilon = 10^{-5}$.

Wszystkie eksperymenty były uruchamiane na komputerze z procesorem Intel Xeon Silver 4116 @ 2.10GHz z 256GB RAM.

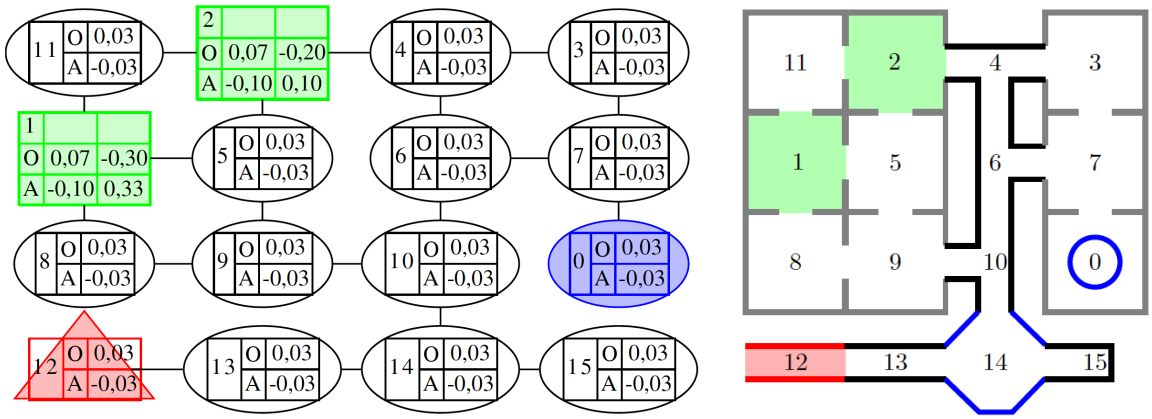
Parametr	**Symbol**	**Wartość**
Wielkość populacji	N	200
Prawdopodobieństwo mutacji	p_m	0.5
Prawdopodobieństwo krzyżowania	p_k	0.8
Presja selekcji	p_s	0.9
Wielkość elity	e	2
Maksymalna liczba pokoleń	g_l	200
Maksymalna liczba pokoleń bez poprawy	g_k	20

Tabela 6.1: Wartości parametrów algorytmu przyjęte w eksperymentach.

6.1 Warehouse Games

Gry Warehouse Games (WHG) (zaproponowane w [33]) inspirowane są praktycznym scenariuszem ochrony magazynów. Rozgrywka toczona jest na nieskierowanym grafie $G = (V, E)$ reprezentującym korytarze i pomieszczenia budynku. Atakujący i Obrońca na początku gry umieszczeni są w określonych z góry wierzchołkach tego grafu. W każdym kroku czasowym gracze mogą albo przemieścić się do dowolnego sąsiedniego (połączonego krawędzią) wierzchołka albo pozostać w obecnym. Zatem strategia prosta gracza g jest listą wierzchołków odwiedzanych w kolejnych krokach czasowych: $\sigma^g = (v_1^g, v_2^g, \ldots, v_m^g)$. W grafie dodatkowo wyróżniony jest podzbiór $T \subseteq V$ wierzchołków nazywanych celami. Przykładową grę przedstawia rysunek 6.1.

Gra kończy się, jeśli zajdzie przynajmniej jeden z poniższych warunków:
(1) gracze znajdą się jednocześnie (w tym samym kroku czasowym) w tym samym wierzchołku v_i ($\exists_{i\in\{1,\ldots,m\}} v_i^A = v_i^O$) - wtedy Atakujący zostaje "złapany", a gracze otrzymują odpowiednie wypłaty związane z tym wierzchołkiem $U_{v_i}^{O+} > 0$ (Obrońca) i $U_{v_i}^{A-} < 0$ (Atakujący),
(2) Atakujący dotrze do któregoś wierzchołka celu $v_i \in T$ i nie zajdzie warunek (1) (nie zostanie "złapany") ($\exists_{i\in\{1,\ldots,m\}} v_i^A \in T \wedge \forall_{j<i}\, v_j^A \neq v_j^O$) - w takim przypadku atak kończy



Rysunek 6.1: Przykładowa gra i odpowiadający jej rzut budynku (za [33]). Cele zaznaczone są prostokątami, wierzchołek startowy Atakującego trójkątem. Obrońca rozpoczyna rozgrywkę z wierzchołka o numerze 0. Liczby w wierzchołkach oznaczają wypłaty poszczególnych graczy.

się sukcesem, a gracze otrzymują wypłaty $U^{O-}_{v_i} < 0$ (Obrońca) i $U^{A+}_{v_i} > 0$ (Atakujący), (3) żadna z sytuacji (1), (2) nie będzie miała miejsca przez m kroków czasowych - wtedy obu graczom przypisywana jest wypłata 0. Strategia prosta Obrońcy i Atakującego polega na wyborze sekwencji wierzchołków, do których będą się przemieszczali w każdym z m kroków czasowych.

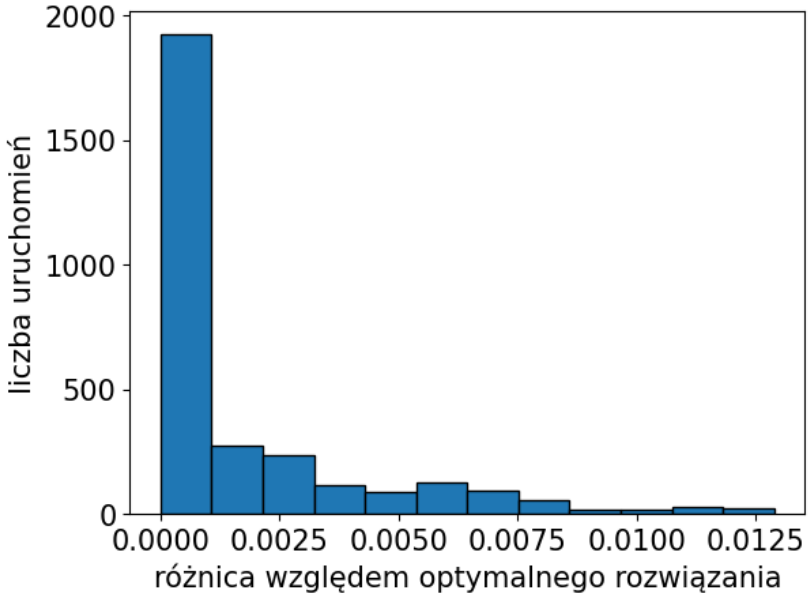
6.1.1 Eksperymenty

Przedstawiony w rozdziale 5 schemat algorytmu EASG nie wymaga żadnych modyfikacji i może wprost być zastosowany do gier WHG. Akcjami, z których składa się strategia prosta Obrońcy, są w tym przypadku identyfikatory wierzchołków w kolejnych krokach czasowych.

Eksperymenty zostały przeprowadzone na 150 grach zaproponowanych w pracy [33]. Każda z nich złożona jest z grafu o 16 wierzchołkach imitującego przestrzeń magazynową 4x4 (jak na rysunku 6.1). Struktura grafu (połączenia między wierzchołkami) została ustalona losowo. Wypłaty graczy przypisane do poszczególnych wierzchołków były z przedziału [-0.67, 0.67]. Testowanych było 6 różnych wartości liczby kroków czasowych (m) - od 3 do 8 (dla każdego kroku czasowego po 25 gier). Wszystkie testowane gry wraz z ich wizualizacjami można pobrać ze strony [45].

6.1.2 Wyniki

Metody dokładne (BC2015 i C2016) w założonym limicie czasu 100h były w stanie policzyć gry co najwyżej 6-krokowe. W sumie było to 100 ze 150 gier testowych. W 72 ze 100 wyżej wymienionych gier algorytm EASG osiągnął wynik optymalny. Średnia różnica między optymalną wypłatą Obrońcy a rezultatem EASG wynosiła 0.0013. Maksymalna różnica wyniosła 0.0127, co stanowi 3.7% możliwego zakresu wypłat (różnicy między maksymalną i minimalną możliwą wypłatą).



Rysunek 6.2: Histogram różnicy wyników zwróconych ze wszystkich uruchomień metody EASG względem rezultatów policzonych algorytmem dokładnym.

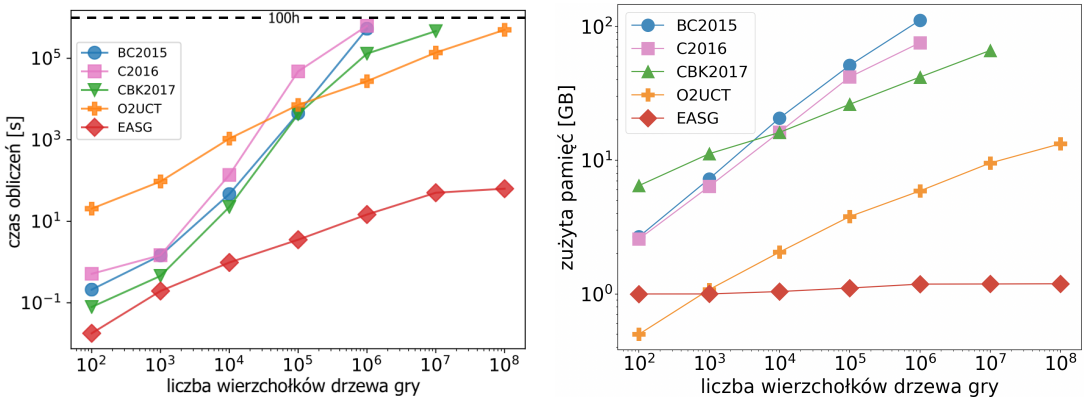
Wyniki otrzymane przez inną metodę przybliżoną O2UCT są porównywalne z EASG - brak statystycznie istotnych różnic zgodnie z testem t-Studenta z progiem istotności p-value < 0.05. O2UCT zwróciło optymalny wynik w przypadku 74 ze 100 gier ze średnią różnicą względem optymalnego wyniku równą 0.0010. Druga z przetestowanych metod przybliżonych - CBK2017 w większej liczbie przypadków była w stanie znaleźć optymalną strategię Obrońcy (77/100), lecz w pozostałych grach różnica względem optymalnego rozwiązania była wyższa, co finalnie przełożyło się na średnią różnicę równą 0.0721. Tabela 6.2 prezentuje porównanie wszystkich 3 metod przybliżonych.

Spośród 30 uruchomień dla wszystkich 150 gier testowych w 45% przypadków odchylenie standardowe wyników zwracanych przez metodę EASG wynosiło 0, co oznacza,

Metoda	Średnia	Max	%
CBK2017	0.072	0.325	**77%**
O2UCT	**0.001**	**0.006**	74%
EASG	**0.001**	0.013	72%

Tabela 6.2: Porównanie 3 metod przybliżonych dla gier WHG w kontekście średniej i maksymalnej różnicy wypłat Obrońcy względem strategii optymalnych oraz odsetka przypadków testowych, dla których dana metoda znalazła optymalne rozwiązanie. Najlepsze wyniki pogrubiono.

że wszystkie uruchomienia algorytmu zwróciły ten sam wynik. Średnie odchylenie standardowe wynosiło 0.0059 z maksymalną wartością 0.1629, co stanowi 36.7% możliwego zakresu wypłat.

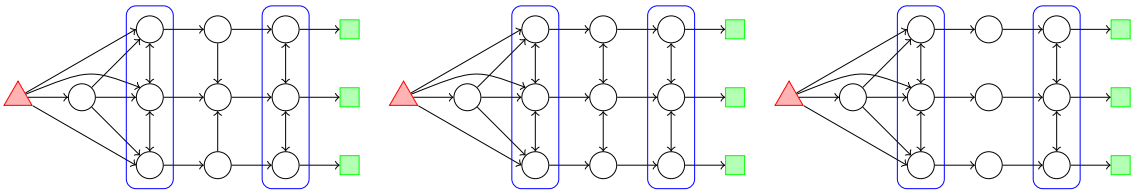


Rysunek 6.3: Porównanie (w skali logarytmicznej) skalowalności czasu obliczeń oraz zużytej pamięci względem liczby wierzchołków drzewa gry dla gier WHG.

Rysunek 6.3 prezentuje porównanie średniego czasu obliczeń oraz ilości zużywanej pamięci dla poszczególnych metod względem liczby wierzchołków w drzewie gry. Czas działania metod EASG i O2UCT skaluje się w przybliżeniu liniowo, podczas gdy pozostałe metody (oparte o programowanie liniowe) wykazują znacznie słabsze właściwości w tym zakresie. Dla najbardziej skomplikowanych gier (z największą liczbą wierzchołków drzewa gry) czas działania algorytmu EASG jest kilkaset razy krótszy niż konkurencyjnych rozwiązań. Dodatkowo algorytm EASG wymaga w przybliżeniu stałej ilości pamięci niezależnie od rozmiaru gry, podczas gry zużycie pamięci pozostałych metod wyraźnie rośnie.

6.2 Search Games

Gry Search Games (SEG) zostały zaproponowane w pracy [9]. Ich charakterystyka przypomina gry WHG bowiem również rozgrywane są na grafie z wyróżnionymi wierzchołkami - celami i gracze w kolejnych krokach czasowych poruszają się między wierzchołkami tego grafu. Jednak w odróżnieniu od gier WHG rozgrywka prowadzona jest na grafie skierowanym, a Obrońca posiada więcej niż jeden zasób, którym może zarządzać (ich liczba będzie oznaczana przez k). Aczkolwiek zasoby te nie mogą poruszać się po grafie dowolnie - każdy z nich ma przydzielony podzbiór wierzchołków, które może odwiedzać (patrz rysunek 6.4).



Rysunek 6.4: Trzy struktury gier SEG użyte w eksperymentach. Trójkątem zaznaczono wierzchołek startowy Atakującego, kwadratami cele, a dwa wyróżnione zbiory 3 wierzchołków to grupy, w ramach których mogą poruszać się dwa zasoby Obrońcy.

Drugą bardzo istotną różnicą względem gier WHG jest częściowa obserwowalność obecności Atakującego. Zakłada się, że Atakujący w każdym odwiedzonym wierzchołku zostawia ślad, który może być (w dowolnym późniejszym kroku czasowym) odkryty przez Obrońcę, jeśli którykolwiek z jego zasobów odwiedzi ten wierzchołek. Z drugiej strony Atakujący ma możliwość zatarcia tych śladów poprzez pozostanie w danym wierzchołku przez więcej niż 1 krok czasowy. Warunki zakończenia gry oraz wypłaty graczy są identyczne jak w przypadku gier WHG: Obrońca dostaje nagrodę (a Atakujący karę) jeśli w którymkolwiek kroku czasowym w tym samym wierzchołku znajdzie się jakikolwiek z jego zasobów oraz Atakujący. Atakujący dostaje nagrodę (a Obrońca karę) w przypadku, gdy Atakującemu uda się dotrzeć do któregokolwiek z celów (i nie zostać w nim złapanym przez zasób Obrońcy). Gra zakończy się zerowymi wypłatami jeśli żadna z powyższych opcji nie zajdzie w określonym limicie kroków czasowych (m). Na rysunku 6.4 zaprezentowano przykładowe gry SEG.

6.2.1 Adaptacja algorytmu EASG

Strategia Obrońcy w grach SEG, ze względu na możliwość odkrywania śladów obecności Atakującego, musi uwzględniać zarówno sytuacje, w których ślad zostanie odkryty jak i brak odkrycia śladów. Ponadto Obrońca może założyć inną strategię dalszego postępowania w przypadku wykrycia śladów w różnych krokach czasowych lub różnych wierzchołkach. Dlatego też bazowa uniwersalna wersja algorytmu EASG musi zostać zmodyfikowana, aby uwzględnić nową postać strategii. W przypadku gier SEG chromosom ma, jak dotychczas, postać strategii prostych i przypisanych im prawdopodobieństw, ale zmienia się definicja strategii prostej. W grach SEG ma ona postać:

$$\sigma^D = \{\mathcal{L}^u_{vs}, u \in D_u, v \in \bar{V}, s \in \{1, \cdots, m\}\} \cup \{\mathcal{L}^u_{\emptyset}, u \in D_u\}, \tag{6.1}$$

gdzie $\mathcal{L}^u_{vs} = (v_1, v_2, \ldots, v_m)$ jest listą wierzchołków odwiedzanych w kolejnych krokach czasowych przez zasób Obrońcy u w przypadku wykrycia śladu obecności Atakującego w wierzchołku v w kroku czasowym s, D_u jest zbiorem zasobów Obrońcy, a $\bar{V} \subseteq V$ podzbiorem wierzchołków grafu osiągalnych przez przynajmniej jeden zasób Obrońcy. $\mathcal{L}^u_{\emptyset}$ jest przypadkiem szczególnym, czyli listą odwiedzanych wierzchołków w przypadku braku wykrycia śladu pozostawionego przez Atakującego. Obrońca wykonuje ruchy zgodnie z aktualnym stanem wiedzy, tzn. odkrycia śladów. Na początku (do momentu wykrycia pierwszego śladu) postępuje zgodnie ze strategią zakodowaną w $\mathcal{L}^u_{\emptyset}$. Następnie, jeśli jednostka u w kroku czasowym s odkryje ślad w wierzchołku v, to od tego momentu (czyli w krokach czasowych $> s$) kolejne ruchy są wykonywane zgodnie z $\mathcal{L}^u_{vs}$. Zasady gry przewidują jedynie odkrycie jednego śladu podczas całej rozgrywki.

Pozostałe operatory algorytmu EASG pozostają niezmienione względem opisanych w rozdziale 5. Mutacja, jak dotychczas, zmienia akcje (w przypadku SEG jest to wybór kolejnych wierzchołków) od losowo wybranego kroku czasowego, lecz w tym wypadku, zamiast losować strategię prostą, w której zajdzie zmiana dodatkowo w ramach strategii losowany jest $\mathcal{L}^u_{vs}$ (lub $\mathcal{L}^u_{\emptyset}$), który zostanie zmodyfikowany. Algorytm EASG z wyżej opisanymi modyfikacjami w dalszej części rozprawy będzie nazywany EASG_{SEG}.

6.2.2 Eksperymenty

W ramach eksperymentów przetestowano 90 gier z $m = 4, 5, 6$ krokami czasowymi (po 30 gier dla każdej wartości m) rozgrywanymi na grafach o 3 różnych strukturach zaproponowanych w [9]. Grafy te zostały zaprezentowane na rysunku 6.4). Kary Obrońcy i Atakującego zostały ustalone dla każdego wierzchołka na -1 ($U_v^{O-} = U_v^{A-} = -1$), a pozostałe wypłaty zostały wylosowane z przedziału $[1, 2]$.

6.2.3 Wyniki

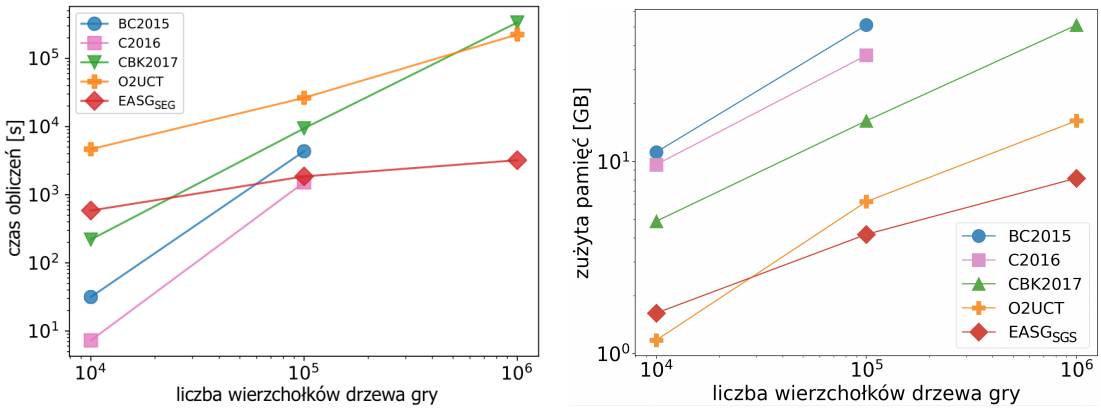
Optymalne wyniki zostały policzone metodami dokładnymi dla 60 z 90 gier testowych (dla kroków czasowych $m \in \{4, 5\}$). Algorytm EASG_{SEG} osiągnął optymalny wynik w 28 przypadkach czyli dla 47% gier. Średnia różnica względem rezultatów optymalnych wyniosła 0.0253 z najwyższą wartością 0.0955 (co stanowi 12.2% zakresu możliwych wypłat). Wyniki te są słabsze niż w przypadku gier WHG. Najprawdopodobniej wynika to z dużo większej przestrzeni przeszukiwań związanej z kodowaniem rozwiązań. Jedna operacja mutacji zamienia tylko jeden element strategii prostej $\mathcal{L} \in \sigma^D$, a nawet w przypadku nieskomplikowanych gier tak reprezentowana strategia prosta składa się z wielu elementów ($m \times k \times |\bar{V}|$). Hipoteza ta została potwierdzona w eksperymentach dodatkowych opisanych w rozdziale 7, gdzie wprowadzenie kilkukrotnego powtarzania operacji mutacji na tym samym chromosomie w ramach jednego pokolenia przyniosło istotną poprawę osiąganych wyników w przypadku gier SEG.

W tabeli 6.3 zaprezentowano porównanie wyników metod przybliżonych. Najlepsze rezultaty pod kątem średniej różnicy względem wyniku optymalnego osiągnął algorytm O2UCT. Metoda EASG_{SEG} znalazła optymalną strategię w największej liczbie przypadków testowych.

Metoda	Średnia	Max	%
CBK2017	0.084	0.276	43%
O2UCT	**0.015**	**0.058**	37%
EASG	0.025	0.096	**47%**

Tabela 6.3: Porównanie 3 metod przybliżonych dla gier SEG w kontekście średniej i maksymalnej różnicy wypłat Obrońcy względem strategii optymalnych oraz odsetka przypadków testowych, dla których dana metoda znalazła optymalne rozwiązanie. Najlepsze wyniki zostały pogrubione.

Rysunek 6.5 przedstawia średni czas obliczeń oraz ilość zużycia pamięci porównywanych metod w zależności od liczby wierzchołków drzewa gry. Dla małych wartości (10^4) krótszy czas działania osiągają metody oparte o programowanie liniowe (BC2015, C2016, CBK2017). Jednak wraz ze wzrostem złożoności gier, przewagę uzyskuje algorytm EASG_{SEG}, którego czas obliczeń rośnie znacznie wolniej niż pozostałych metod, a co za tym idzie, pozwala on na rozwiązywanie większych gier o bardziej skomplikowanej strukturze. Również ilość zużywanej pamięci przez algorytm ewolucyjny jest mniejsza niż w przypadku konkurencyjnych metod, a przewaga ta uwidacznia się szczególnie w przypadku większych gier.



Rysunek 6.5: Porównanie (w skali logarytmicznej) skalowalności czasu obliczeń oraz zużytej pamięci względem liczby wierzchołków drzewa gry dla gier SEG.

6.3 FlipIt Games

Kolejnym rodzajem gier rozważanych w rozprawie są gry FlipIt Games (FIG) [75]. Inspiracją do ich powstania był scenariusz cyberbezpieczeństwa, w którym Atakujący próbuje przejąć pewne elementy infrastruktury sieciowej (np. komputery, rutery), a Obrońca może podejmować akcje, które pozwalają na odzyskanie kontroli nad tymi zasobami.

Gra rozgrywana jest na grafie skierowanym, przez pewną z góry określoną liczbę kroków czasowych m. W kolejnych krokach czasowych każdy z graczy podejmuje akcję polegającą na wybraniu jednego wierzchołka, nad którym będzie próbował przejąć kontrolę (ang. *flip the node*). Przez v_s^g będziemy oznaczać wierzchołek, którego pró-

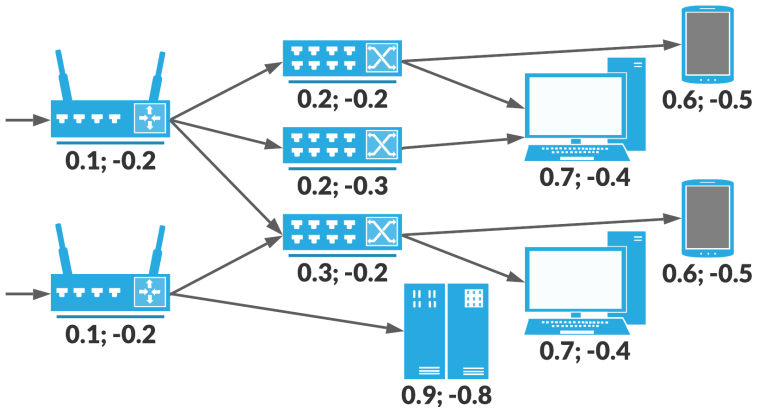
bę przejęcia podejmuje gracz g w kroku czasowym s. Na początku rozgrywki tylko niektóre z wierzchołków (określane jako wierzchołki wejściowe) są dostępne dla Atakującego - odpowiada to scenariuszowi, w którym tylko część infrastruktury sieciowej jest publicznie dostępna. Atakujący rozpoczyna infiltrację sieci od jednego z tych wierzchołków. Operacja przejmowania przez gracza kontroli nad wierzchołkiem kończy się sukcesem jedynie w przypadku, gdy co najmniej jeden z poprzedników (wierzchołków, z których krawędź skierowana jest do przejmowanego wierzchołka) znajduje się pod kontrolą tego gracza lub przejmowany jest jeden z wierzchołków wejściowych grafu (bez poprzedników). W sytuacji gdy obaj gracze próbują w tym samym kroku czasowym przejąć kontrolę nad tym samym wierzchołkiem, pozostaje on w posiadaniu tego gracza, do którego należał wcześniej. Na początku gry wszystkie wierzchołki są kontrolowane przez Obrońcę.

Każdemu wierzchołkowi przypisane są dwie wartości: nagroda za posiadanie nad nim kontroli ($U_v^+ > 0$) oraz koszt próby przejęcia kontroli (niezależnie od tego czy jest udana, czy nie) ($U_v^- < 0$). Wypłata graczy liczona jest w sposób nieco inny niż w poprzednich rodzajach gier. W FIG wypłatą gracza jest suma nagród ze wszystkich kontrolowanych wierzchołków w kolejnych krokach czasowych (jeśli wierzchołek jest kontrolowany przez wiele kroków czasowych, to nagroda za niego liczona jest wielokrotnie) pomniejszona o sumę kosztów wszystkich prób przejęcia kontroli (niezależnie czy były one udane, czy nie). Wartość wypłaty gracza g można wyrazić wzorem:

$$U_g = \sum_{s\in\{1,\ldots,m\}} \sum_{v\in R_s(g)} U_v^+ + \sum_{s\in\{1,\ldots,m\}} U_{v_s^g}^-, \tag{6.2}$$

gdzie $R_s(g)$ jest podzbiorem wierzchołków kontrolowanych przez gracza g po kroku czasowym s. Rysunek 6.6 przedstawia przykładową grę FIG.

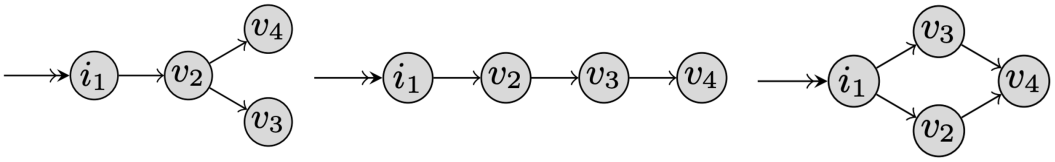
Należy zauważyć, że z założenia gracze w trakcie rozgrywki nie wiedzą, które z wierzchołków są przez nich kontrolowane (wariant *No-Info* [14]), a swoją strategię (polegającą na wyborze dla każdego z kroków czasowych wierzchołka, nad którym będzie próba przejęcia kontroli) planują na początku i nie zmieniają jej w miarę postępu rozgrywki.



Rysunek 6.6: Wizualizacja przykładowego scenariusza z obszaru cyberbezpieczeństwa zamodelowanego jako gra FIG. Wierzchołki grafu są reprezentowane przez elementy infrastruktury sieciowej - rutery, centrum danych, komputery osobiste i urządzenia mobilne. Każdy z nich ma przypisane dwie liczby: nagrodę za posiadanie nad nim kontroli oraz koszt próby jego przejęcia. Dwa elementy po lewej stronie są wierzchołkami wejściowymi sieci.

6.3.1 Eksperymenty

Ze względu na złożoność gier FIG oraz chęć porównania z konkurencyjnymi metodami w eksperymentach wykorzystano 60 gier rozgrywanych na 3 relatywnie nieskomplikowanych strukturach grafów zaproponowanych w pracy [14] i przedstawionych na rysunku 6.7. Przetestowano 4 różne wartości liczby kroków czasowych $m \in \{3, 4, 5, 6\}$.



Rysunek 6.7: Przetestowane struktury grafów gier FIG - za [14].

Przedstawiona w rozdziale 5 bazowa wersja algorytmu nie wymaga żadnych istotnych zmian czy adaptacji do gier FIG. Strategie proste zakodowane są w postaci listy wierzchołków, których próba przejęcia kontroli będzie podejmowana w kolejnych krokach czasowych: $(v_1^g, v_2^g, \ldots, v_m^g)$. Operatory mutacji, krzyżowania, selekcji oraz warunek stopu pozostają bez zmian. Jedynie proces ewaluacji z oczywistych powodów musi być dostosowany do reguł gry i sposobu liczenia wypłat wyrażonego wzorem 6.2.

6.3.2 Wyniki

Metody dokładne w założonym limicie czasu 100h zwróciły wyniki dla 45 z 60 gier testowych. W przypadku 33 z nich (73%) algorytm ewolucyjny (EASG) zwrócił tę samą optymalną wypłatę Obrońcy. Średnia różnica względem rozwiązania optymalnego dla wszystkich 45 gier wyniosła 0.0087, z maksymalnym wynikiem 0.0321 (co stanowi 6.4% zakresu wypłat tj. różnicy między maksymalną i minimalną możliwą wartością wypłaty Obrońcy).

Tabela 6.4 przedstawia porównanie metod przybliżonych. Najlepszą metodą pod względem liczby gier, dla których zwrócony został wynik optymalny jest algorytm EASG. O2UCT osiągnął najniższą średnią oraz maksymalną różnicę względem rozwiązania optymalnego. Nie stwierdzono statystycznie istotnej różnicy wyników na podstawie testu t-Studenta z progiem istotności p-value < 0.05.

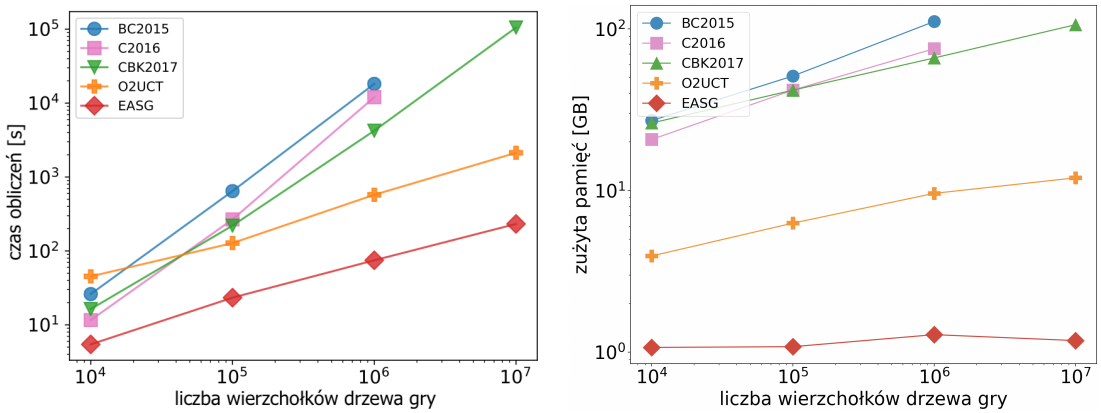
Metoda	Średnia	Max	%
CBK2017	0.014	0.127	63%
O2UCT	**0.004**	**0.012**	58%
EASG	0.009	0.032	**73%**

Tabela 6.4: Porównanie 3 metod przybliżonych dla gier FIG w kontekście średniej i maksymalnej różnicy wypłat Obrońcy względem strategii optymalnych oraz odsetka przypadków testowych, dla których dana metoda znalazła optymalne rozwiązanie. Najlepsze wyniki pogrubiono.

Rysunek 6.8 demonstruje czas obliczeń oraz zużywaną pamięć wszystkich porównywanych metod. Podobnie jak w przypadku wcześniej omawianych gier (WHG i SEG) metody oparte o programowanie liniowe skalują się wyraźnie gorzej niż algorytmy EASG i O2UCT. We wszystkich przypadkach czas działania EASG jest kilkukrotnie niższy niż pozostałych metod. Ilość zużywanej pamięci przez algorytm EASG jest w przybliżeniu stała.

6.4 Games on a Plane

Wcześniej opisywane gry rozgrywane były w przestrzeni dyskretnej, tzn. akcjami Obrońcy i Atakującego w poszczególnych krokach czasowych był wybór wierzchołków ze skończonego zbioru możliwych opcji. Gry na płaszczyźnie (ang. *Security Games on a Plane*)



Rysunek 6.8: Porównanie (w skali logarytmicznej) skalowalności czasu obliczeń oraz zużytej pamięci względem liczby wierzchołków drzewa gry dla gier FIG.

(w dalszej części nazywane w skrócie SGP) powiększają ilość możliwych akcji obu graczy, ponieważ są rozgrywane w przestrzeni ciągłej. Charakterystycznym elementem jest też fakt, że nie tylko gracze, ale również cele poruszają się (na płaszczyźnie). Model tych gier został zaproponowany przez autora rozprawy i innych współautorów w publikacji [37]. Inspiracją do powstania gier SGP był scenariusz ochrony promów turystycznych na Morzu Śródziemnym. Model gry zakłada zbiór T (o liczności n) ruchomych celów (promów) i k jednakowych zasobów Obrońcy, które w praktyce mogą być przykładowo szybszymi niż promy jednostkami wodnymi (np. motorówkami). Ponadto, w grze zdefiniowany jest zbiór punktów P (portów), które są miejscami startu i destynacji. Każdy z celów $t \in T$ kursuje naprzemiennie między dwoma ustalonymi punktami (portami) $p_1^t \in P, p_2^t \in P$. Cele poruszają się między tymi punktami w linii prostej ze stałą prędkością v według określonego harmonogramu H^t. Harmonogram określa momenty wyruszenia celu t z przypisanych mu portów: $H^t = (r_1^t, r_2^t, \ldots, r_{l^t}^t)$, gdzie r_i^t jest chwilą (czasem, który upłynął od chwili 0) wyruszenia celu t z punktu p_1^t dla $i \bmod 2 = 0$ lub z punktu p_2^t dla $i \bmod 2 = 1$ (cele poruszają się pomiędzy punktami naprzemiennie). l^t jest długością harmonogramu i dla różnych celów może przyjmować różne wartości. Dodatkowo $\forall_{i<j} r_i^t < r_j^t$ oraz różnica momentów startów z kolejnych punktów jest większa niż czas pokonania drogi między nimi: $\forall_{i \in \{1,\ldots,l^t-1\}} r_{i+1}^t - r_i^t \geqslant |p_1^t p_2^t| v$, gdzie $|p_1^t p_2^t|$ oznacza odległość Euklidesową między punktami p_1^t i p_2^t. Harmonogram dopuszcza sytuację, że przez jakiś czas cel nie porusza się, będąc w jednym z punktów p_1^t, p_2^t.

Cel jest *chroniony* jeśli w odległości co najwyżej r od niego znajduje się co najmniej

jedna jednostka Obrońcy. Na początku gry każda z jednostek Obrońcy jest przypisana do jednego z punktów z P i stamtąd rozpoczyna rozgrywkę. Gra składa się z m kroków czasowych o długości τ każdy. Początek każdego z kroków czasowych jest punktem decyzyjnym (czyli są to kolejno chwile $0, \tau, 2\tau, \ldots, (m-1)\tau$).

Obrońca w kolejnych punktach decyzyjnych wybiera dla każdej swojej jednostki dowolny punkt na płaszczyźnie, do którego dana jednostka będzie płynąć. Punkt ten musi być osiągalny w kolejnym kroku czasowym, czyli odległość do niego (z aktualnej pozycji) nie może być większa niż $v_{max}\tau$, gdzie v_{max} jest maksymalną prędkością każdej z jednostek Obrońcy.

Atakujący wybiera krok czasowy, w którym rozpocznie atak oraz jeden z celów ($t \in T$). Atak trwa τ_A czasu. Jeśli w którymkolwiek momencie trwania ataku (niekoniecznie w wyróżnionych punktach decyzyjnych) atakowany cel będzie chroniony (w odległości co najwyżej r od niego znajdzie się jakaś jednostka Obrońcy), to atak jest nieudany (gracze otrzymują wypłaty $U_t^{O+} > 0$ i $U_t^{A-} < 0$). W przeciwnym przypadku atak kończy się sukcesem (gracze otrzymują wypłaty $U_t^{O-} < 0$ i $U_t^{A+} > 0$). Atakujący może w ogóle nie podejmować ataku - wtedy gra kończy się zerowymi wypłatami.

6.4.1 Adaptacja algorytmu EASG

Kodowanie rozwiązań

Postać zakodowanych w chromosomie strategii mieszanych jest analogiczna jak w przypadku bazowej wersji EASG - jest to zbiór strategii prostych wraz z przypisanymi im prawdopodobieństwami. Każda ze strategii prostych zapisana jest jako lista punktów w kolejnych krokach czasowych każdej z jednostek: $\sigma^D = \{(d_0^1, d_1^1, \ldots, d_m^1), \ldots, (d_0^k, d_1^k, \ldots, d_m^k)\}$, gdzie d_j^i oznacza współrzędne punktu położenia i-tej jednostki Obrońcy na koniec j-tego kroku czasowego (w chwili $j\tau$).

Na początku każdej jednostce i przypisywany jest punkt startowy, będący losowo wybranym portem: $d_0^i \in P$.

Następnie populacja początkowa ustalana jest losowo, tzn. dla każdej jednostki i, po kolei dla każdego kroku czasowego wybierane jest losowe położenie jednostki d_j^i spośród wszystkich możliwych do osiągnięcia, czyli $\forall_{s \in \{1,\ldots,m\}} |d_{s-1}^i d_s^i| \leqslant \tau v_{max}$.

Mutacja

Mutacja polega na zamianie położenia losowej jednostki w losowym kroku czasowym. Nowe położenie musi spełniać warunki poprawności strategii, tzn. musi zapewnić możliwość dotarcia do wybranego punktu w czasie τ (długość kroku czasowego) z poprzedniego położenia i analogicznie dotarcia do następnego położenia. W praktyce losowanie nowego położenia jednostki i w kroku czasowym j następuje z części wspólnej kół o promieniu $v_{max}\tau$ i środkach w punktach d^i_{j-1} oraz d^i_{j+1}.

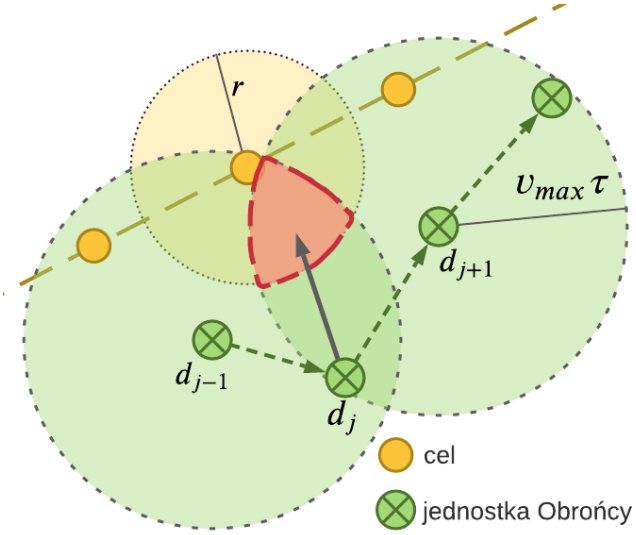
Funkcja przystosowania

W trakcie eksperymentów zauważono, że wartość oczekiwanej wypłaty Obrońcy (jak w bazowej wersji EASG) nie jest wystarczającym kryterium oceny osobników. W takim przypadku między innymi nie są rozróżnialne sytuacje, w których strategia Obrońcy powoduje (a) brak ochrony celów w jakimkolwiek kroku czasowym oraz (b) ochronę wszystkich celów we wszystkich krokach czasowych oprócz jednego (w którym żaden cel nie jest chroniony). W obu tych przypadkach oczekiwana wypłata jest jednakowa (gdyż Atakujący może wybrać do ataku krok czasowy bez ochrony), ale strategią "lepszą" ("bliższą" optymalnej) jest ta druga. Z tego względu wprowadzono dodatkowy mechanizm rozstrzygania remisów. W tym celu liczona jest suma wypłat Obrońcy (grającego strategią zakodowaną w chromosomie) dla każdego kroku czasowego przy założeniu, że Atakujący zaatakuje w tym kroku, tzn. $\sum_{i=0}^{m} U^O(t_x, s_i)$, $t_x = \arg\max_t U^A(t, s_i), t \in T$, gdzie $U^G(t, s_i)$ jest wypłatą gracza $G \in \{O, A\}$ w przypadku ataku na cel t w i-tym kroku czasowym. Im większa jest ta suma, tym lepiej przystosowany jest osobnik i istnieje większe prawdopodobieństwo jego promocji do następnego pokolenia. Mechanizm ten stosowany jest jedynie do porównywania przystosowania osobników w przypadku równości podstawowego kryterium, czyli oczekiwanej wypłaty Obrońcy.

Lokalna optymalizacja

Krok lokalnej optymalizacji jest uruchamiany (dla każdego osobnika) po operacjach mutacji i krzyżowania, przed selekcją. Polega on na próbie poprawy wszystkich strategii prostych zakodowanych w chromosomie w następujący sposób. Dla każdej jednostki Obrońcy iteracyjnie w kolejnych krokach czasowych poszukiwane jest nowe położenie,

które zwiększy liczbę chronionych celów przy jednoczesnym zachowaniu pozostałych położeń w niezmienionej formie. Analogicznie jak w przypadku mutacji nowe możliwe położenie jednostki i w kroku j musi być w odległości nie większej niż $v_{max}\tau$ od punktów w poprzednim d^i_{j-1} oraz kolejnym kroku czasowym d^i_{j+1}. Wizualizacja przykładowej lokalnej optymalizacji przedstawiono na rysunku 6.9.



Rysunek 6.9: Wizualizacja przykładowej procedury lokalnej optymalizacji. Na rysunku zaznaczono położenie celu oraz jednostki Obrońcy na koniec kolejnych kroków czasowych. W wyniku optymalizacji położenie jednostki Obrońcy d_j zamieniane jest na losowy punkt z obszaru zaznaczonego na czerwono, tak aby jednostka ochraniała przedstawiony cel.

Eksperymenty wykazały, że zaproponowany krok lokalnej optymalizacji jest kluczowym elementem działania algorytmu. Bez niego otrzymywane wyniki były średnio mniejsze o 38% i w żadnym przypadku nie przewyższyły wyników algorytmu z lokalną optymalizacją. Lokalna optymalizacja zapewnia, że położenia jednostek Obrońcy są bardziej skupione wokół celów. Algorytm w ten sposób szybciej znajduje strategie z wysokimi wypłatami Obrońcy zamiast losowo "błądzić" po całej płaszczyźnie.

Warunek stopu, operatory krzyżowania i selekcji pozostały niezmienione względem bazowego rozwiązania opisanego w rozdziale 5. Algorytm EASG z wyżej opisanymi modyfikacjami w dalszej części będzie nazywany EASG_{SGP}.

6.4.2 Eksperymenty

W eksperymentach użyto 6150 losowych gier testowych wygenerowanych zgodnie z następującymi parametrami: liczba celów $1 \leqslant n \leqslant 50$, liczba kroków czasowych $m \in \{2, 4, 6, 8, 10\}$, liczba jednostek Obrońcy $k \in \{max(1, n-2), \ldots, n+1\}$, liczba punktów startowych (portów) $2\sqrt{n} \leqslant |P| \leqslant 3\sqrt{n}$, $r = 1.0$, $\nu_{max} = 1.0$, $\tau_A = \tau$.

Ze względu na specyfikę gry nie jest możliwe wprost zastosowanie metod, z którymi wcześniej porównywany był algorytm EASG. Dlatego do porównań użyto następujących metod:

DOBBS [58] - algorytm dokładny oparty o technikę programowania liniowego i całkowitoliczbowego opisany w rozdziale 4.1.1. Aby użycie tego algorytmu było możliwe, gra najpierw musi zostać zmodyfikowana do postaci dyskretnej. W oryginalnym sformułowaniu gry przestrzeń możliwych strategii prostych Obrońcy jest ciągła - zarządza on położeniami jednostek na płaszczyźnie. Jednak z punktu widzenia wypłat graczy istotna jest jedynie informacja, ile celów w danym położeniu jest chronionych. Dlatego też, zamiast rozważać położenia jednostek na płaszczyźnie, w zredukowanej wersji gry rozważane będą jedynie pokrycia celów. Obrońca decyduje, które z celów będą chronione w poszczególnych krokach czasowych bez zdefiniowania wprost w jaki sposób będą chronione, tzn. jak będą poruszały się jego jednostki. Oryginalna przestrzeń strategii Obrońcy jest zastępowana możliwymi do zrealizowania sekwencjami ochrony celów w czasie. Procedura zamiany przestrzeni decyzyjnej Obrońcy jest kosztowna obliczeniowo i działa w czasie wykładniczym względem liczby celów oraz jednostek Obrońcy. Pozwala jednak uzyskać grę w formie dyskretnej, do której rozwiązania można użyć wcześniej przytoczonego algorytmu DOBBS.

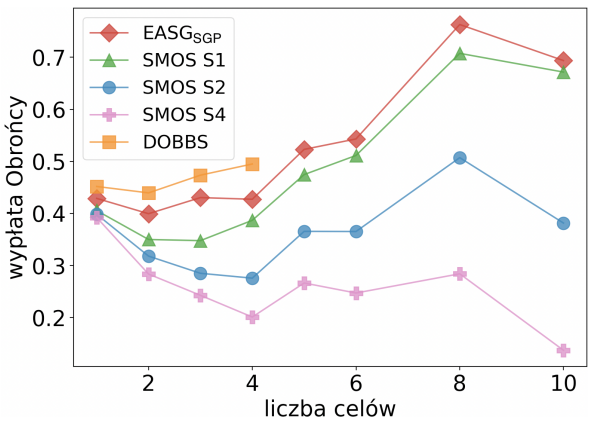
SMOS [37] - metoda będąca rozszerzeniem algorytmu z pracy [79] na gry sekwencyjne o sumie niezerowej. Jest oparta o stopniowo rozszerzany zestaw ograniczeń liniowych oraz reguły łączące grupy podobnych strategii, co pozwala przyspieszyć działanie algorytmu w zamian za utratę gwarancji uzyskania wyniku optymalnego. W celu redukcji liczby rozważanych strategii Obrońcy metoda SMOS dyskretyzuje przestrzeń poprzez podział płaszczyzny na jednakowe kwadraty. W ten sposób agreguje pewne obszary, traktując wszystkie punkty wewnątrz nich jednakowo. W zależności od wielkości

agregowanych obszarów (długości boku kwadratu) zmienia się stopień utraty jakości wyników, a tym samym zysk kosztu obliczeniowego. Zaprezentowane będą 3 warianty algorytmu S1 (największy koszt obliczeniowy), S2 oraz S4 (największa utrata jakości wyników). Tak zredukowana gra jest następnie rozwiązywana przy pomocy techniki programowania liniowego i całkowitoliczbowego w sposób podobny do metody DOBBS. Szczegóły dotyczące działania tego algorytmu przedstawiono w pracy [37].

6.4.3 Wyniki

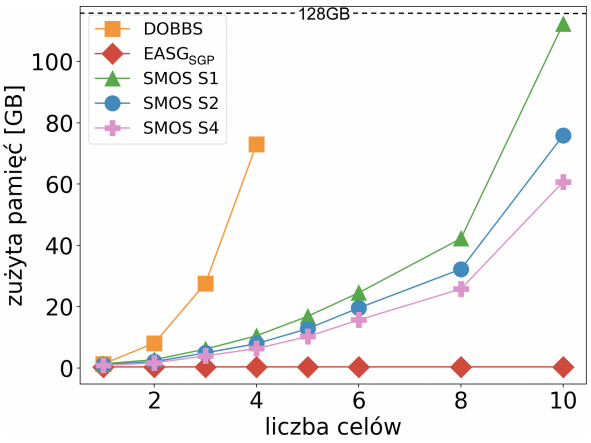
Spośród wszystkich 6150 gier algorytm dokładny (DOBBS) był w stanie policzyć jedynie 605 o maksymalnie 4 krokach czasowych, 3 celach i 2 jednostkach Obrońcy (dla większych gier przekraczany był limit pamięci wynoszący 128GB). Zaproponowany algorytm EASG_{SGP} zwrócił optymalny wynik w przypadku 599 (97%) spośród tych gier oraz był w stanie policzyć wszystkie 6150 gier testowych. Algorytm SMOS S4 (najmniej kosztowny obliczeniowo wariant SMOS) policzył 3146 gier. Podobnie jak w przypadku algorytmu DOBBS ograniczeniem dla większych gier okazała się ilość dostępnej pamięci, na której zapotrzebowanie w przypadku gier o więcej niż 10 celach przekraczało maksymalny limit 128GB.

Rysunek 6.10 prezentuje porównanie średnich wyników (oczekiwanych wypłat Obrońcy) w zależności od liczby celów. Algorytmy EASG_{SGP} oraz SMOS S1 zwracają rezultaty o podobnej jakości, zbliżonej do wyników optymalnych. Dla pozostałych algorytmów SMOS S2 i S4 wyniki są słabsze i wraz ze wzrostem skomplikowania gry (liczbą celów) przewaga metod EASG_{SGP} i SMOS S1 nad nimi powiększa się.



Rysunek 6.10: Średnia wypłata Obrońcy w zależności od liczby celów.

Średnie odchylenie standardowe wyników zwracanych przez EASG_{SGP} wynosiło $3 \cdot 10^{-3}$ i tylko w 554 (9%) przypadkach testowych było większe niż 10^{-5}.



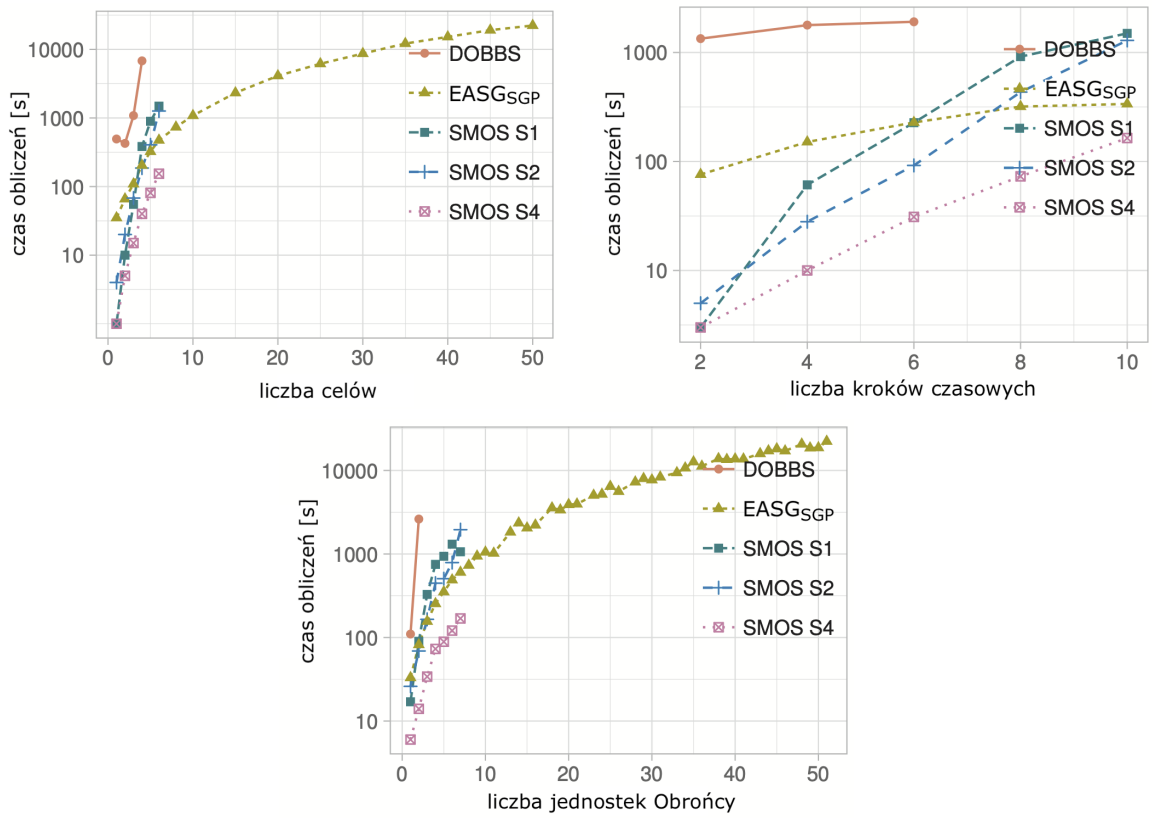
Rysunek 6.11: Średnie zużycie pamięci w zależności od liczby celów.

Na rysunku 6.11 zaprezentowano średnie zużycie pamięci przez poszczególne metody. Najgorzej pod tym względem skaluje się algorytm DOBBS, dla którego już w przypadku 5 celów przekraczany był limit pamięci ustalony na 128GB. Podejście ewolucyjne EASG_{SGP} wykazuje w przybliżeniu stałe zużycie pamięci niezależnie od liczby celów, ponieważ wielkość zakodowanych w chromosomach rozwiązań nie zależy od wielkości gry (liczby celów, kroków czasowych czy wielkości płaszczyzny, na której odbywa się rozgrywka).

Rysunek 6.12 przedstawia czas obliczeń poszczególnych metod w zależności od głównych parametrów gier - liczby celów, liczby kroków czasowych i jednostek Obrońcy. Dla małych wartości tych parametrów EASG_{SGP} czasami ma dłuższy czas działania niż metoda SMOS, jednak w miarę gdy gra staje się bardziej skomplikowana, proponowany algorytm ewolucyjny skaluje się wyraźnie lepiej niż pozostałe metody.

6.5 Signaling Games

Kolejny typ gier należy do popularnej w ostatnich latach tematyki nazywanej Green Security Games [26, 24]. Gry z tej rodziny modelują sytuacje związane z ochroną środowiska naturalnego, np. walka z kłusownictwem [25] czy nielegalnym połowem ryb [28]. Rozpatrywany w tym rozdziale rodzaj gier został zainspirowany rzeczywistym scenariuszem ochrony parków narodowych w Afryce [5]. Wykorzystuje się tam drony, które



Rysunek 6.12: Porównywanie skalowalności czasu obliczeń w zależności od liczby celów, liczby kroków czasowych oraz liczby jednostek Obrońcy.

regularnie patrolują wybrane części parku oraz wyposażone są w kamery termowizyjne i system automatycznej detekcji anomalii. W momencie gdy dron wykryje zagrożenie, może przesłać informację o tym zagrożeniu do centrali (gdzie zgłoszenie zostanie zweryfikowane przez ludzi i w podejrzane miejsce zostanie wysłany strażnik) lub wysłać sygnał dźwiękowy i świetlny w celu odstraszenia kłusowników. Tego rodzaju gry nazywane są grami obronnymi z sygnalizacją (ang. *Security Games with Signaling*), w skrócie SGS.

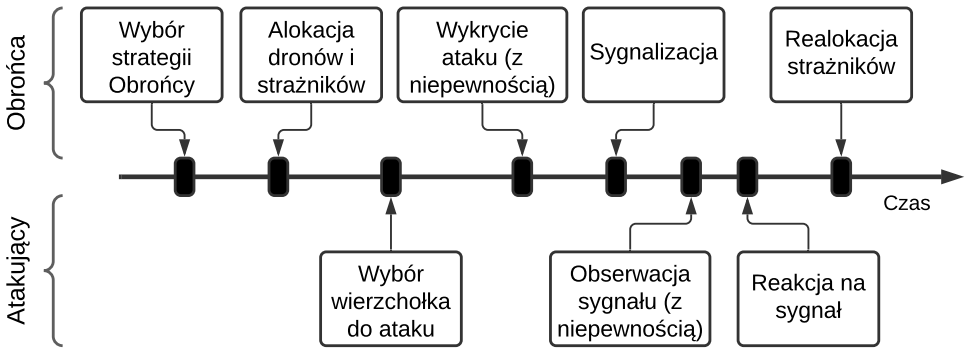
Charakterystyczną cechą gier SGS jest rozróżnienie zasobów Obrońcy na dwie kategorie: drony i strażników, które pełnią odmienne role i mają różne możliwości. Dodatkowym wyróżnikiem tych gier jest uwzględnianie niepewności związanej z systemem automatycznej detekcji zagrożenia przez drony. W praktyce zdarza się, że system jest w pewnym stopniu zawodny i nie zawsze potrafi wykryć zagrożenie.

Formalnie rozważana jest gra między Obrońcą a Atakującym na grafie $G = (V, E)$ o n wierzchołkach. Każdy z wierzchołków $t \in V$ stanowi potencjalny cel i związane są

z nim 4 wypłaty $U_t^{O+} > 0 > U_t^{O-}$ i $U_t^{A+} > 0 > U_t^{A-}$ (patrz rozdział 3.3). Obrońca ma do dyspozycji k_d dronów i k_s strażników (ludzi).

Gra składa się z kilku etapów. Najpierw Obrońca wybiera swoją strategię, według której będzie dalej postępował. Zgodnie z tą strategią k_d dronów i k_s strażników przypisuje do wybranych wierzchołków grafu - celów. W kolejnej fazie Atakujący (zgodnie z założeniem gier Stackelberga), znając strategię Obrońcy, wybiera jeden wierzchołek t, który atakuje. Jeśli do zaatakowanego wierzchołka przypisany był strażnik to "łapie" on Atakującego, gracze otrzymują odpowiednie wypłaty (U_t^{O+}, U_t^{A-}), a gra kończy się. W przeciwnym przypadku następuje etap reakcji Obrońcy. Może on wykryć atak przy pomocy drona (jeśli znajdował się on w zaatakowanym wierzchołku). Dron może wysłać jeden z dwóch sygnałów: tzw. słaby sygnał s_0, który polega jedynie na wysłaniu informacji o ataku (lub o jego braku) do strażników lub sygnał silny s_1, który oprócz informacji przekazanej strażnikom, powoduje uruchomienie sygnałów świetlnych i dźwiękowych, które mają na celu odstraszenie Atakującego. Jeśli strażnicy otrzymają od drona informację o ataku i istnieje strażnik przypisany do wierzchołka sąsiadującego z atakowanym, to zostaje on przemieszczony do atakowanego wierzchołka i udaremnia działania Atakującego. Gra kończy się odpowiednimi wypłatami (U_t^{O+}, U_t^{A-}). Jeśli Atakujący zauważy sygnał s_1 wysyłany przez drona, może zrezygnować z ataku. Wtedy obaj gracze otrzymują wypłatę równą 0. Ostatnią fazą jest realokacja strażników. Każdy strażnik przemieszcza się do jednego z sąsiadujących wierzchołków zgodnie z ustaloną na początku strategią Obrońcy. Jeśli któryś z wierzchołków, do którego przemieszczani są strażnicy, był wierzchołkiem zaatakowanym (i Atakujący nie zrezygnował z ataku), to Obrońca "łapie" Atakującego, a graczom przydzielane są odpowiednie wypłaty (U_t^{O+}, U_t^{A-}). W przypadku gdy nie zaszła żadna z powyższych sytuacji, w której Obrońca udaremnił działania Atakującego, atak uznaje się za zakończony sukcesem i graczom przypisywane są wypłaty U_t^{O-} i U_t^{A+}. Rysunek 6.13 przedstawia kolejność etapów gry i czynności wykonywanych przez graczy. Z kolei rysunek 6.14 obrazuje wszystkie możliwe ścieżki prowadzące do rozstrzygnięcia gry.

Istotną własnością gier SGS jest uwzględnianie niepewności. W grze wyróżnione są dwa jej rodzaje: *niepewność wykrycia* i *niepewność obserwacji*. Niepewność wykrycia wynika z ograniczeń systemu dronów rozpoznającego zagrożenia. Rozpatrywane są tyl-



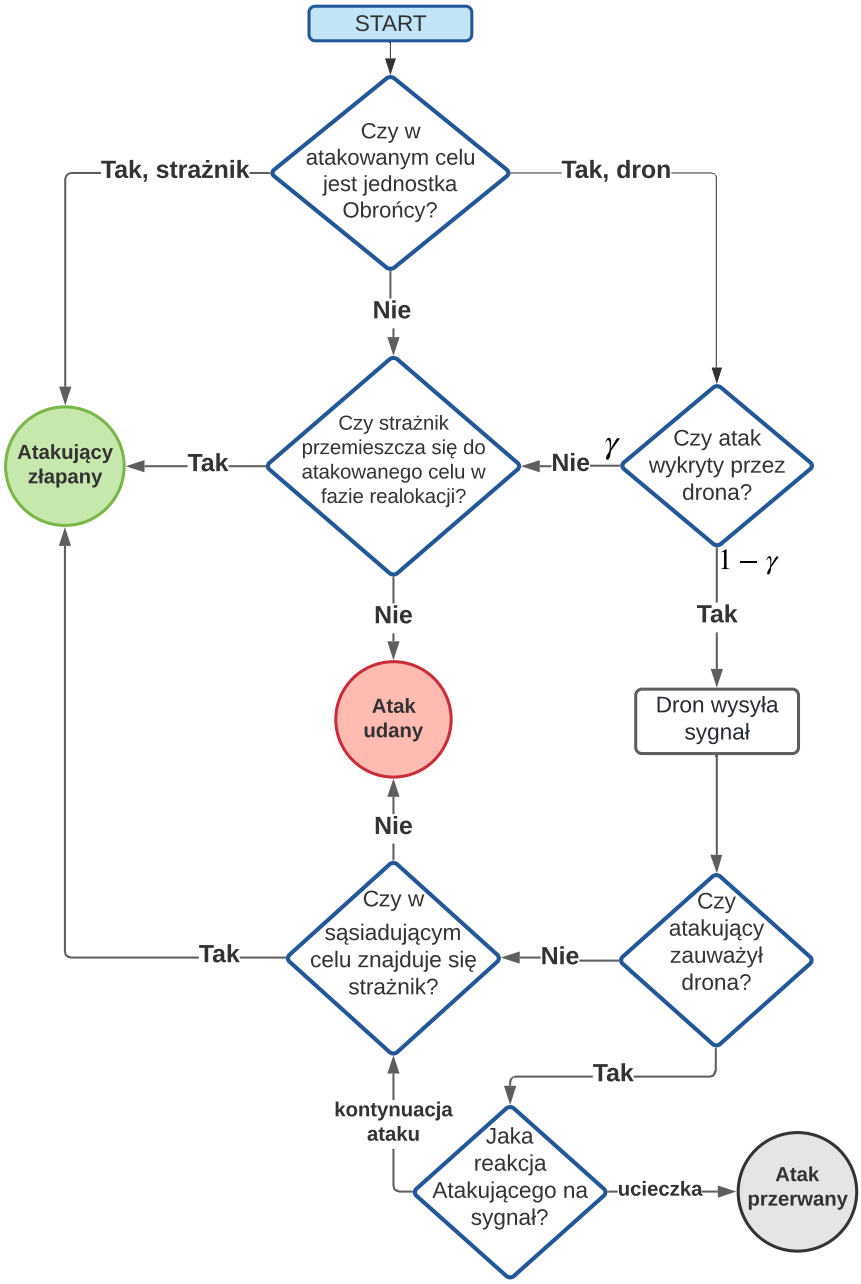
Rysunek 6.13: Kolejność czynności wykonywanych przez graczy w grach SGS.

ko sytuacje, w których dron nie wykryje niebezpieczeństwa, podczas gdy ono będzie miało miejsce. Odwrotna sytuacja (brak zagrożenia sygnalizowany przez dron jako niebezpieczeństwo) jest w praktyce rozwiązywana przez czynnik ludzki - operator dronów ma dostęp do podglądu obrazu wideo i może zweryfikować daną sytuację manualnie. Przez γ będziemy oznaczać prawdopodobieństwo braku wykrycia zdarzenia w sytuacji jego wystąpienia.

Druga z rozpatrywanych niepewności jest związana z możliwymi błędami obserwacji otoczenia przez Atakującego i jego błędną interpretacją sygnałów. Wprowadźmy macierz $\mathbf{\Omega}$, która będzie określała prawdopodobieństwa warunkowe $P[x|y]$ zaobserwowania przez Atakującego sygnału y w sytuacji wysłania sygnału x :

$$\Omega = \begin{bmatrix} P[\bar{s}|\bar{s}] & P[\bar{s}|s_0] & P[\bar{s}|s_1] \\ P[s_0|\bar{s}] & P[s_0|s_0] & P[s_0|s_1] \\ P[s_1|\bar{s}] & P[s_1|s_0] & P[s_1|s_1] \end{bmatrix} \tag{6.3}$$

gdzie $\bar{s}$ oznacza brak drona w wierzchołku, s_0 oznacza obecność drona, który wysyła sygnał słaby - informację o stanie zagrożenia do strażników bez żadnych dodatkowych widocznych zewnętrznie działań, s_1 oznacza obecność drona wysyłającego sygnał silny, czyli oprócz informacji jak s_0 są uruchamiane sygnały świetlne i dźwiękowe w celu odstraszenia Atakującego. Obserwacja przez Atakującego sygnału s_0 oznacza, że zauważył on obecność drona mimo braku uruchomienia przez niego sygnałów świetlnych i dźwiękowych. Zgodnie z sugestią przedstawioną w pracy [5] niektóre z powyższych sytuacji nie mają odzwierciedlenia w rzeczywistości i ich prawdopodobieństwo zawsze



Rysunek 6.14: Diagram obrazujący zasady gier obronnych z sygnalizacją (SGS).

wynosi 0, np. Atakujący nie zaobserwuje sygnału s_1 w przypadku braku obecności drona w wierzchołku ($\bar{s}$). Dlatego też rozważana będzie tylko ograniczona postać macierzy Ω sparametryzowana przez κ:

$$\Omega_\kappa = \begin{bmatrix} 1 & \kappa & \frac{\kappa}{2} \\ 0 & 1-\kappa & \frac{\kappa}{2} \\ 0 & 0 & 1-\kappa \end{bmatrix}$$

Wprowadzona niepewność powoduje konieczność rozszerzenia strategii Obrońcy, który musi uwzględnić wszystkie możliwe sytuacje. Na strategię prostą Obrońcy składa się alokacja k_d dronów do wybranych wierzchołków-celów, alokacja k_s strażników do celów, strategia realokacji, czyli wybór k_s wierzchołków sąsiadujących (w grafie) z wybranymi alokacjami strażników. Dodatkowo strategia Obrońcy musi zawierać też plan sygnalizacji dronów, tzn. dla każdego drona określone jest prawdopodobieństwo wysłania sygnałów s_0 i s_1 w przypadkach wykrycia ataku oraz braku wykrycia ataku (jak pokazano w [5] z uwagi na niepewność wykrycia czasami warto wysłać sygnał nawet w przypadku braku zaobserwowanego zagrożenia). Ponadto w planie sygnalizacji powinna również zostać uwzględniona obecność (lub jej brak) strażnika w którymś z sąsiednich wierzchołków - intuicyjnie, jeśli w sąsiednim wierzchołku jest strażnik, który może zareagować i dotrzeć do Atakującego, to korzystniejsze jest wysłanie sygnału słabego i nieostrzeganie przeciwnika o nadchodzącym strażniku niż próba przepłoszenia Atakującego.

Strategia Atakującego składa się tylko z dwóch elementów: atakowanego wierzchołka oraz planu reakcji na zaobserwowane sygnały, tzn. decyzji czy kontynuować atak w przypadku dostrzeżenia obecności drona (sygnał s_0) albo jego sygnalizacji świetlnej/dźwiękowej (sygnał s_1).

Bardziej szczegółowy opis gier SGS oraz dyskusję ich właściwości można znaleźć w [5].

6.5.1 Adaptacja algorytmu EASG

Ze względu na rozbudowaną strukturę gry i szerszy niż w poprzednich przypadkach zbiór elementów decyzyjnych strategii graczy, algorytm ESGS musi być odpowiednio zaadaptowany. Opisane poniżej zmiany zachowują zaproponowane wcześniej podstawowe zasady działania algorytmu i są tylko jego dostosowaniem do nowych elementów, takich jak wprowadzenie niepewności czy różnych rodzajów zasobów Obrońcy.

Kodowanie rozwiązań

Rozwiązanie (strategia Obrońcy) jest kodowane w chromosomie jako lista strategii prostych z ich prawdopodobieństwami oraz strategia sygnalizacji dronów:

$$CH_q = \{(\sigma_1^q, p_1^q), \ldots, (\sigma_i^q, p_i^q), \ldots, (\sigma_{l_q}^q, p_{l_q}^j), \boldsymbol{\Psi}_{\boldsymbol{q}}^{\boldsymbol{\theta}}, \boldsymbol{\Phi}_{\boldsymbol{q}}^{\boldsymbol{\theta}}\},$$

gdzie l_q jest liczbą strategii prostych w chromosomie, σ_i^q jest strategią prostą z przypisanym jej prawdopodobieństwem $p_i^q \in [0,1]$, $\sum_{i=1}^{l_q} p_i^q = 1$.
$\theta \in \{\bar{\mu}, \mu^+, \mu^-\}$ reprezentuje następujące możliwe stany, w których może znajdować się dron umieszczony w wierzchołku t:
$\bar{\mu}$ - brak strażnika w sąsiednim wierzchołku (nie ma strażnika, który mógłby zareagować na sygnał),
μ^+ - istnieje strażnik, który w fazie realokacji ma zaplanowane przemieszczenie się do wierzchołka t,
μ^- - brak strażnika, który w fazie realokacji ma zaplanowane przemieszczenie się do wierzchołka t, ale w przynajmniej jednym wierzchołku sąsiadującym z t jest strażnik (który może zareagować na otrzymany sygnał i przemieścić się do t).
$\boldsymbol{\Psi}_{\boldsymbol{q}}^{\boldsymbol{\theta}} = [\Psi_{q,1}^{\theta}, \Psi_{q,2}^{\theta}, \ldots, \Psi_{q,n}^{\theta}]$ jest strategią sygnalizacji w przypadku wykrycia ataku, tj. $\Psi_{q,\nu}^{\theta}$ jest warunkowym prawdopodobieństwem, że dron w wierzchołku ν wyśle sygnał s_0 pod warunkiem, że znajduje się on w stanie alokacji θ i **wykrył** atak. Analogicznie $\boldsymbol{\Phi}_{\boldsymbol{q}}^{\boldsymbol{\theta}} = [\Phi_{q,1}^{\theta}, \Phi_{q,2}^{\theta}, \ldots, \Phi_{q,n}^{\theta}]$ jest strategią sygnalizacji w przypadku niewykrycia ataku, tj. $\Phi_{q,\nu}^{\theta}$ jest warunkowym prawdopodobieństwem, że dron w wierzchołku ν wyśle sygnał s_0 pod warunkiem, że znajduje się on w stanie alokacji θ i **nie wykrył** ataku. Dron wysyła albo sygnał s_0, albo sygnał s_1, więc prawdopodobieństwo wysłania sygnału s_1 jest dopełnieniem prawdopodobieństwa wysłania sygnału s_0: $1 - \Psi_{q,\nu}^{\theta}$ (dla wykrycia ataku) lub $1 - \Phi_{q,\nu}^{\theta}$ (dla braku wykrycia ataku).

Zgodnie z definicją gier SGS opisana w poprzednim rozdziale strategia prosta σ jest reprezentowana przez trójelementową tuplę: $\sigma = (V_s, V_d, V_r)$, gdzie
$V_s = \{\nu_1^s, \nu_2^s, \ldots, \nu_{k_s}^s\} \subseteq V$ jest zbiorem wierzchołków, w których znajdują się strażnicy,
$V_d = \{\nu_1^d, \nu_2^d, \ldots, \nu_{k_d}^d\} \subseteq V$ jest zbiorem wierzchołków, w którym umieszczone są drony,
$V_r = \{\nu_1^r, \nu_2^r, \ldots, \nu_{k_s}^r\} \subseteq V$ jest planem realokacji, tzn. zbiorem wierzchołków (sąsiadu-

jących z odpowiednimi wierzchołkami z V_s), gdzie przemieszczą się strażnicy w fazie realokacji (w przypadku braku sygnalizacji drona o ataku w sąsiednim wierzchołku) - i-ty strażnik przemieszcza się z wierzchołka v_i^s do v_i^r jeśli odpowiednia krawędź w grafie istnieje. Jeśli wierzchołki v_i^s i v_i^r nie są połączone krawędzią, strażnik w fazie realokacji pozostaje w v_i^s.

Populacja początkowa

Populacja początkowa składa się z losowych jednoelementowych strategii mieszanych (dla każdej z nich niezależnie wybierany jest losowy podzbiór wierzchołków do alokacji strażników i dronów oraz realokacji strażników) oraz strategii sygnalizacji, w której wszystkie prawdopodobieństwa są przypisane losowo (jednostajnie z przedziału $[0; 1]$).

Mutacja

W podstawowej wersji algorytmu jedynie losowo wybrane strategie proste podlegały mutacji. W przypadku gier SGS struktura chromosomów jest bogatsza, a mutacje powinny dotyczyć wszystkich jej elementów tak, aby przy pomocy operatorów ewolucyjnych było możliwe uzyskanie dowolnej strategii Obrońcy. Dlatego też w SGS istnieją dwa typy mutacji: mutacja alokacji (M_1) oraz sygnalizacji (M_2). Mutacja alokacji wprowadza zaburzenie do losowo wybranej strategii prostej σ. Najpierw wybierany jest losowo element σ, który będzie mutowany, tj. V_s, V_d lub V_r. A następnie losowy wierzchołek z tego zbioru v_{old} jest zamieniany na losowy inny v_{new}. $M_1((V_s, V_d, V_r)) = (V_s', V_d', V_r')$ oraz $\exists!_{z\in\{s,d,r\}} V_z \neq V_z' : V_z = (v_1, \ldots, v_{old}, \ldots, v_k)$, $V_z' = (v_1, \ldots, v_{new}, \ldots, v_k)$. Drugi typ mutacji wybiera losowe prawdopodobieństwo ze strategii sygnalizacji ($\boldsymbol{\Psi}^{\boldsymbol{\theta}}$ lub $\boldsymbol{\Phi}^{\boldsymbol{\theta}}$, $\theta \in \{\bar{\mu}, \mu^+, \mu^-\}$)), a następnie zamienia je na jego dopełnienie: $M_2(\boldsymbol{\Psi}^{\boldsymbol{\theta}}) = \boldsymbol{\Psi}^{\boldsymbol{\theta\prime}}$ oraz $\exists!_{z\in\{1,\ldots,n\}}$: $\boldsymbol{\Psi}^{\boldsymbol{\theta}} = [\Psi_1^\theta, \ldots, \Psi_z^\theta, \ldots, \Psi_n^\theta]$, $\boldsymbol{\Psi}^{\boldsymbol{\theta\prime}} = [\Psi_1^\theta, \ldots, 1 - \Psi_z^\theta \ldots, \Psi_n^\theta]$ (dla $\boldsymbol{\Phi}^{\boldsymbol{\theta}}$ zapis mutacji wygląda analogicznie).

Krzyżowanie

Rezultatem krzyżowania chromosomów CH_1 i CH_2 jest

$$CH_{1\text{-}2} = \{(\sigma_1^1, \frac{p_1^1}{2}), (\sigma_2^1, \frac{p_2^1}{2}), \ldots, (\sigma_{l_1}^1, \frac{p_{l_1}^1}{2}), (\sigma_1^2, \frac{p_1^2}{2}), (\sigma_2^2, \frac{p_2^2}{2}), \ldots, (\sigma_{l_2}^2, \frac{p_{l_2}^2}{2}), \boldsymbol{\Psi}_{\mathbf{1\text{-}2}}^{\boldsymbol{\theta}}, \boldsymbol{\Phi}_{\mathbf{1\text{-}2}}^{\boldsymbol{\theta}}\},$$

gdzie

$\boldsymbol{\Psi^{\theta}_{1\text{-}2}} = [\frac{1}{2}(\Psi^{\theta}_{1,1} + \Psi^{\theta}_{2,1}), \ldots, \frac{1}{2}(\Psi^{\theta}_{1,n} + \Psi^{\theta}_{2,n})]$, $\boldsymbol{\Phi^{\theta}_{1\text{-}2}} = [\frac{1}{2}(\Phi^{\theta}_{1,1} + \Phi^{\theta}_{2,1}), \ldots, \frac{1}{2}(\Phi^{\theta}_{1,n} + \Phi^{\theta}_{2,n})]$.

Po tej operacji z chromosomu $CH_{1\text{-}2}$ każda ze strategii σ_i^j jest usuwana z prawdopodobieństwem $(1 - p_i^j)^2$, a następnie prawdopodobieństwa wszystkich pozostałych strategii są normalizowane, aby sumowały się do 1.

Lokalna optymalizacja

Procedura inicjalizacji populacji, mutacja i krzyżowanie działają w sposób losowy i wygenerowane przez nie strategie mogą być niedopuszczalne lub w oczywisty sposób nieefektywne. W celu ich poprawy wprowadzony został dodatkowy krok lokalnej optymalizacji, który uruchamiany jest dla każdego osobnika po zastosowaniu mutacji i krzyżowania.

Na początku procedura ta sprawdza, czy wszystkie zakodowane strategie są dopuszczalne, tzn. czy dla każdego wierzchołka z V_s i odpowiadającego mu wierzchołka realokacji z V_r istnieje krawędź w grafie, na którym rozgrywana jest gra $(\forall_{i \in \{1,\ldots,k_s\}}(\nu_i^p, \nu_i^r) \in E)$. Jeśli dla i-tej pary wierzchołków ten warunek nie jest spełniony, to wierzchołek ν_i^r zamieniany jest na losowy wierzchołek sąsiadujący z ν_i^p.

W drugiej kolejności sprawdzane jest, czy istnieje wierzchołek, do którego przypisany jest więcej niż jeden zasób Obrońcy (dron lub strażnik). W takim przypadku "nadmiarowy" zasób jest przenoszony do innego losowo wybranego wierzchołka bez przypisanego zasobu (jeśli jest to strażnik to dodatkowo zmieniany jest jego wierzchołek realokacji zgodnie z procedurą opisaną w poprzednim akapicie).

Ewaluacja

Ewaluacja rozwiązań zakodowanych w osobnikach jest przeprowadzana poprzez znalezienie najlepszej strategii Atakującego i policzenie dla niej wypłat graczy. Najlepsza strategia Atakującego wybierana jest poprzez sprawdzenie wszystkich możliwych strategii prostych. W przypadku gier SGS składają się one z dwóch elementów: atakowanego wierzchołka oraz reakcji (kontynuacja ataku lub ucieczka) na zaobserwowany sygnał - s_0 (obecność drona bez widocznej sygnalizacji) lub s_1 (widoczna sygnalizacja drona). Wypłaty graczy liczone są zgodnie z definicją gry, uwzględniając niepewność

wykrycia i obserwacji. Na ich podstawie liczone są prawdopodobieństwa wszystkich możliwych ścieżek zakończenia gry (patrz rysunek 6.14), wypłaty w każdym z tych stanów mnożone są przez prawdopodobieństwa ich osiągnięcia i sumowane.

Warunek stopu oraz operator selekcji pozostały niezmienione względem rozwiązania opisanego w rozdziale 5. Algorytm EASG z wyżej opisanymi modyfikacjami w dalszej części będzie nazywany EASG_{SGS}.

6.5.2 Eksperymenty

Zbiory danych

Mimo że rozważane gry są silnie inspirowane praktycznym scenariuszem ochrony zasobów naturalnych w Afryce Południowej, to dostęp do rzeczywistych danych jest utrudniony. Strażnicy nie chcą publicznie ujawniać, gdzie znajdują się najcenniejsze zasoby, na przykład miejsca występowania gatunków zagrożonych wyginięciem. Dlatego też w testach użyto 342 sztucznie wygenerowane gry, których sposób tworzenia bazował na publicznie dostępnych danych - np. zakresy wypłat odpowiadały szacunkowej wartości kości słoniowej na czarnym rynku (nagroda Atakującego) oraz średnim przychodom z ekoturystyki, które mogą zostać utracone przez działania kłusowników (kara Obrońcy). Szerszą dyskusję na temat praktycznych inspiracji można znaleźć w [5] oraz [83].

Gry zostały podzielone na 4 grupy w zależności od typu grafów, na których są rozgrywane: *rzadkie* (50 gier) z deg_{avg}[1]$= 2$, *pośrednie* (50 gier) z $deg_{avg} = \frac{n}{2}$, *gęste* (50 gier) z $deg_{avg} = n - 2$ oraz *lokalnie gęste* (192 gry), które składały się z połączonych klik różnej wielkości. Grafy rzadkie, pośrednie i gęste generowane były przy pomocy modelu Wattsa-Strogatza [81] z liczbą wierzchołków $n \in \{10, 20, \ldots, 100\}$ (po 5 gier dla każdego rozmiaru). Grafy lokalnie gęste złożone były z n_c klik o rozmiarze c każda ($n_c, c \in \{3, 4, \ldots, 10\}$) zgodnie z jedną z poniższych reguł łączenia $r \in \{1, 2, 3\}$:

$r = 1$: dokładnie jeden wierzchołek z każdej kliki połączony z dokładnie dwiema sąsiednimi[2] klikami,

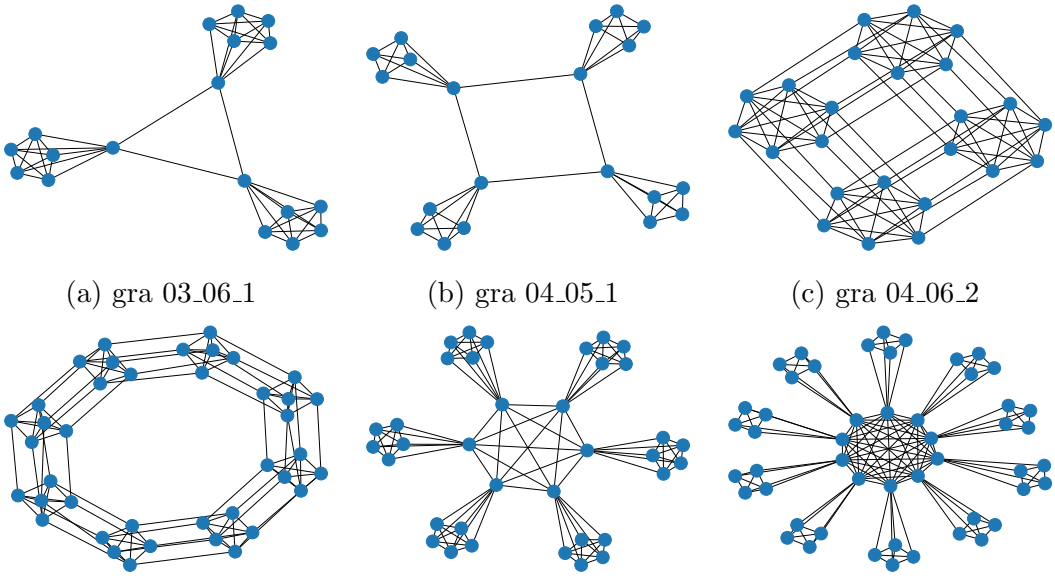
$r = 2$: każdy wierzchołek z kliki połączony z odpowiadającymi mu wierzchołkami z dwóch sąsiednich klik,

[1] deg_{avg} oznacza średni stopień wierzchołka w grafie

[2] dla każdej kliki przypisane są dwie kliki sąsiadujące, wszystkie kliki tworzą cykl klik

$r = 3$: dokładnie jeden wierzchołek z każdej kliki połączony z wybranymi wierzchołkami wszystkich innych klikami.

Każda gra oparta o grafy lokalnie gęste jest identyfikowana przez $n_c\_c\_r$. Przykładowo 05_03_1 oznacza grę z grafem złożonym z 5 klik o wielkości 3 połączonych regułą 1. Rysunek 6.15 pokazuje 6 przykładów grafów lokalnie gęstych użytych w testach.



(a) gra 03_06_1 (b) gra 04_05_1 (c) gra 04_06_2

(d) gra 08_05_2 (e) gra 06_06_3 (f) gra 10_05_3

Rysunek 6.15: Przykłady grafów lokalnie gęstych używanych w zbiorze testowym.

Liczba strażników i dronów w grach testowych zależała od liczby wierzchołków grafu (n) i była ustalona jako $k_s = \sqrt{\frac{n}{2}}$ i $k_d = \frac{2}{3}n - k_s$. Parametry niepewności γ i κ były losowane dla każdej gry niezależnie z rozkładem jednostajnym z przedziału $[0, 1]$.

Porównywane metody

W literaturze zaproponowano dotychczas kilka metod rozwiązujących gry SGS. Tutaj zostanie przedstawiony jedynie ich ogólny zarys, bardziej szczegółowy opis każdej z metod można znaleźć w cytowanych pracach.

SELP [5] - jest to algorytm dokładny oparty o serię programów liniowych zaadoptowanych do SGS z [17]. Każdy program odpowiada jednemu wierzchołkowi grafu $t \in V$ i oblicza on maksymalną wypłatę Obrońcy przy założeniu, że atak nastąpi w wierzchołku t. Następnie wybierana jest najwyższa z tak otrzymanych wypłat. Każdy z programów liniowych ma złożoność wykładniczą i przyjmuje następującą postać:

$$\max_{x,p,\Phi,\Psi} U_{\bar{s}}^{O}(t) + U_{s_0}^{O}(t) + U_{s_1}^{O}(t), \text{ takie że} \tag{6.4}$$

$$\sum_{\sigma \in \Sigma_O : \sigma_v = \theta} p_\sigma = x_v^\theta \quad \forall \theta \in \{\bar{\mu}, \mu^+, \mu^-\}, \forall v \in V \tag{6.5}$$

$$\sum_{\sigma \in \Sigma_O} p_\sigma = 1 \tag{6.6}$$

$$p_\sigma \geqslant 0 \quad \forall \sigma \in \Sigma_O \tag{6.7}$$

$$U_{\bar{s}}^{A}(t) + U_{s_0}^{A}(t) + U_{s_1}^{A}(t) \geqslant U_{\bar{s}}^{A}(v) + U_{s_0}^{A}(v) + U_{s_1}^{A}(v) \quad \forall v \neq t \tag{6.8}$$

$$0 \leqslant \Phi_v^\theta \leqslant x_v^\theta \quad \forall \theta \in \{\bar{\mu}, \mu^+, \mu^-\}, \forall v \in V \tag{6.9}$$

$$0 \leqslant \Psi_v^\theta \leqslant x_v^\theta \quad \forall \theta \in \{\bar{\mu}, \mu^+, \mu^-\}, \forall v \in V \tag{6.10}$$

$U_s^G(v)$ oznacza oczekiwaną wypłatę gracza G podczas ataku na wierzchołek v w przypadku sygnalizacji s (patrz równanie 6.3), p_σ to prawdopodobieństwo zagrania strategii σ w strategii mieszanej Obrońcy, x_v^θ jest prawdopodobieństwem, że wierzchołek v jest w stanie θ (patrz rozdział 6.5.1), Ψ i Φ stanowią strategię sygnalizacji jak w rozdziale 6.5.1. Równania 6.5-6.7 zapewniają poprawność strategii mieszanej Obrońcy, natomiast równanie 6.8 gwarantuje, że najlepszym wyborem do ataku (maksymalizującym wypłatę Atakującego) jest wierzchołek t.

SBP [5] - jest to algorytm przybliżony będący modyfikacją poprzedniego podejścia. Metoda ta przy użyciu techniki *branch-and-price* [29] redukuje złożoność problemu, przyspieszając tym samym obliczenia. Algorytm na początku rozwiązuje (przy pomocy techniki programowania liniowego i całkowitoliczbowego) problem ze znacznie zredukowaną przestrzenią strategii Obrońcy $\Sigma'_O \subseteq \Sigma_O$, a następnie poszukuje takiej strategii prostej $\sigma \in \Sigma_O \setminus \Sigma'_O$, której dodanie do rozważanego zbioru strategii Σ'_O spowoduje zwiększenie oczekiwanej wypłaty Obrońcy. W kolejnym kroku zbiór Σ'_O jest rozszerzany o tę strategię, a problem rozwiązywany w oparciu o nowy zbiór strategii Obrońcy. Procedura jest powtarzana do momentu, w którym znalezienie σ podwyższającej wypłatę Obrońcy jest niemożliwe. W pracy [5] dodatkowo zaproponowano modyfikację tego algorytmu, w której losowy wybór zbioru początkowego Σ'_O zastąpiono heurystyką dobierającą obiecujące strategie początkowe na podstawie ich wypłat. Ten wariant

oznaczany będzie przez **SBP+W**.

m-CombSGPO [77] - to algorytm przybliżony oparty o technikę uczenia ze wzmocnieniem wykorzystujący sieć *Multi-agent Deep Q-Network* [76]. Każda z jednostek Obrońcy (strażnicy i drony) jest traktowana jako agent, którego działaniem jest wybór wierzchołka w grafie oraz decyzja o sygnalizacji. System uczony jest zgodnie z koncepcją *Centralized Training and Decentralised Execution* [44], która zakłada, że agenci podejmują decyzję o swoich akcjach niezależnie, ale informacja zwrotna o jakości podjętych akcji jest liczona globalnie na podstawie działań wszystkich agentów. Ocenę akcji agentów stanowi jakość przyjętej strategii Obrońcy wyliczonej na podstawie podjętych decyzji. Szczegóły działania metody m-CombSGPO można znaleźć w [77].

Wyżej wymienione metody będą porównane z zaproponowanym algorytmem ewolucyjnym EASG_{SGS}.

6.5.3 Wyniki

Algorytm dokładny (SELP) był w stanie policzyć jedynie gry z maksymalnie 20 celami (wierzchołkami). Dla większych gier przekraczany był limit pamięci wynoszący 128GB. Zatem porównanie z wynikami optymalnymi było możliwe jedynie dla 30 przykładów testowych: po 6 gier rzadkich, pośrednich i gęstych oraz 12 lokalnie gęstych. Wyniki porównania przedstawione są w tabeli 6.5. Natomiast w tabeli 6.6 przedstawiono średnie wypłaty osiągnięte na wszystkich 342 instancjach testowych.

	SBP	SBP+W	m-CombSGPO	EASG_{SGS}
rzadkie	**0.00** | **0.00** | **6/6**	**0.00** | **0.00** | **6/6**	10.45 | 0.14 | 0/6	0.90 | 0.01 | 1/6
pośrednie	15.77 | 0.17 | 1/6	7.92 | 0.09 | **2/6**	45.12 | 0.42 | 0/6	**1.51** | **0.02** | 1/6
gęste	18.93 | 0.26 | **0/6**	9.25 | 0.09 | **0/6**	42.52 | 0.57 | **0/6**	**1.98** | **0.03** | **0/6**
lokalnie gęste	13.45 | 0.28 | 1/12	3.16 | 0.05 | **4/12**	31.34 | 0.52 | 0/12	**1.42** | **0.02** | 3/12

Tabela 6.5: Porównanie metod z wynikami optymalnymi. Pierwsza liczba oznacza średnią różnicę między rozwiązaniem dokładnym a zwróconym przez daną metodę. Druga liczba to stosunek tej różnicy do wartości rozwiązania dokładnego. Ostania wartość jest ułamkiem instancji testowych, dla których dana metoda znalazła rozwiązanie optymalne (różnica mniejsza niż $\varepsilon = 0.001$).

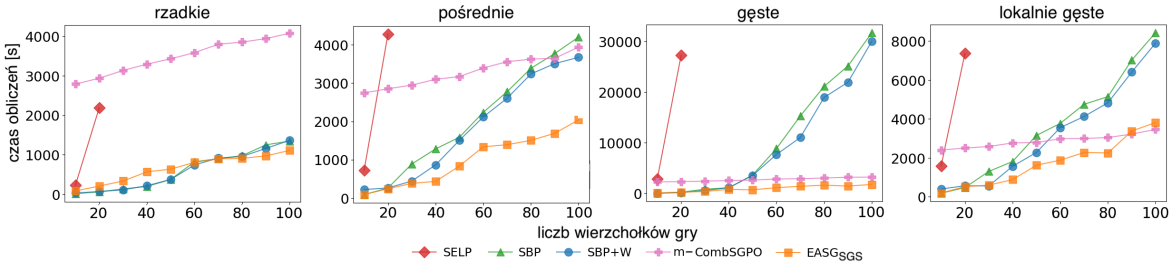
Zaprezentowane wyniki pokazują, że dla gier z grafami rzadkimi najlepsze wyniki zwraca metoda SBP+W. Dla pozostałych typów gier EASG_{SGS} osiąga wyniki lepsze od wszystkich porównywanych metod. Przyczyną jest wzrost złożoności obliczeniowej

	SBP	SBP+W	m-CombSGPO	EASG_{SGS}
rzadkie	-86.68 (84%)	**-86.01 (92%)**	-419.86 (0%)	-91.32 (6%)
pośrednie	-75.01 (2%)	-72.75 (36%)	-255.73 (0%)	**-69.92 (62%)**
gęste	-58.72 (2%)	-57.98 (34%)	-149.14 (0%)	**-51.47 (64%)**
lokalnie gęste	-60.68 (4%)	-57.80 (26%)	-340.65 (0%)	**-54.36 (70%)**

Tabela 6.6: Porównanie średnich wypłat Obrońcy dla wszystkich gier testowych. W nawiasie podany jest odsetek gier, dla których dana metoda osiągnęła najlepszy wynik (w niektórych przypadkach kilka metod osiągnęło ten sam wynik, dlatego wartości te nie muszą sumować się do 100%).

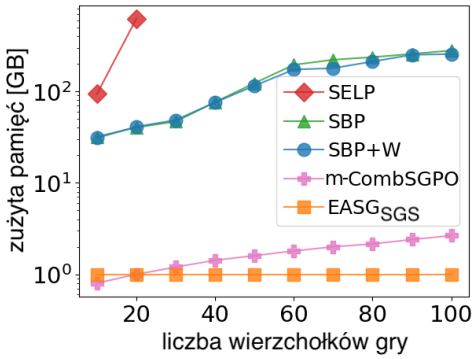
w przypadku zwiększenia liczby połączeń między wierzchołkami - każde dodatkowe połączenie zwiększa liczbę możliwości realokacji i zwiększa przestrzeń przeszukiwań. W sumie, w przypadku 200 z 342 gier testowych metoda EASG_{SGS} osiągnęła najlepszy wynik. W przypadku wszystkich typów gier przewaga zwycięskiej metody jest statystycznie istotna zgodnie z jednostronnych testem t-Studenta z progiem istotności p-value < 0.05 (normalność rozkładu zmiennej zależnej została potwierdzona testem Shapiro-Wilka). Średnie odchylenie standardowe metody EASG_{SGS} wyniosło 0.86 (co stanowi około 1.2% średniej wypłaty) z maksymalną wartością 1.63.

Rysunek 6.16 przedstawia porównanie czasów działania poszczególnych metod. Metoda dokładna (SELP) ma wykładniczy czas działania. Pozostałe metody w przybliżeniu skalują się liniowo względem liczby wierzchołków. W przypadku metod SBP oraz SBP+W widać znaczny wzrost czasu działania wraz ze wzrostem gęstości grafów. Algorytm EASG_{SGS} we wszystkich przypadkach odznacza się najlepszym czasem działania.



Rysunek 6.16: Czas działania w zależności od liczby wierzchołków gry.

Rysunek 6.17 przedstawia skalowalność poszczególnych metod pod kątem zużywanej pamięci RAM. Widać wyraźną różnicę między metodami opartymi o programowanie liniowe (SELP, SBP, SBP+W), które zużywają znaczne zasoby pamięciowe nawet dla



Rysunek 6.17: Ilość zużywanej pamięci (skala logarytmiczna) w zależności od liczby wierzchołków gry.

stosunkowo niewielkich gier, a pozostałymi algorytmami (m-CombSGPO, $EASG_{SGS}$). Ze względu na swoją specyfikę metoda $EASG_{SGS}$ niezależnie od typu gry oraz liczby wierzchołków grafu zużywa w przybliżeniu stałą ilość pamięci (około 200MB), ponieważ niezależnie od warunków gry przechowuje populację o tym samym rozmiarze i podobnej strukturze.

6.6 Podsumowanie

Zaproponowana metoda ewolucyjna została przetestowana na szerokim zbiorze gier o różnej charakterystyce, wielkościach i zasadach. Część z gier była rozgrywana na grafach w przestrzeni dyskretnej (WHG, SEG, FIG), inne na płaszczyźnie w przestrzeni ciągłej (SGP). Niektóre z gier zakładały posiadanie przez Obrońcę jednej jednostki (WHG, FIG), inne wymagały podejmowania decyzji dotyczącej wielu jednostek o jednakowej (SGP) lub różnej (SGS) charakterystyce. Pewne gry uwzględniały częściową obserwowalność (SEG) lub niepewność działań graczy (SGS). We wszystkich przypadkach zaproponowany schemat algorytmu ewolucyjnego był możliwy do zastosowania przy niewielkich modyfikacjach. Świadczy to o uniwersalności przyjętego podejścia i możliwości jego prostej adaptacji do różnego rodzaju gier.

W większości przypadków algorytm ewolucyjny powtarzalnie osiągał wyniki optymalne lub bliskie optymalnym. Jakość zwracanych rezultatów nie odbiegała znacząco od tych uzyskiwanych przez inne metody opisywane w literaturze, które niekiedy były stworzone i zoptymalizowane pod konkretny rodzaj gry. Należy podkreślić, że dla

wszystkich typów gier osiągnięte wyniki zostały policzone z tym samym uniwersalnym zestawem wartości parametrów. Parametry algorytmu nie były w żadnym stopniu dostosowane do rodzaju gry, jej wielkości czy poziomu skomplikowania.

Istotną przewagą proponowanej metody jest jej skalowalność czasowa i pamięciowa. Dzięki zużyciu mniejszej ilości zasobów obliczeniowych, czas działania algorytmu jest zauważalnie niższy niż konkurencyjnych metod, co uwidacznia się szczególnie w przypadku gier o większych rozmiarach (większej liczbie kroków czasowych, wierzchołków w grafie, jednostek Obrońcy itp.). W przeciwieństwie do większości innych metod, algorytm ewolucyjny nie buduje drzewa gry, dzięki czemu wykorzystuje znacznie mniej zasobów pamięciowych. Ich zużycie jest w przybliżeniu stałe niezależnie od rozmiaru rozwiązywanej gry.

Powyższe cechy świadczą o skuteczności zaproponowanego rozwiązania i możliwości jego skutecznego zastosowania do szerokiej klasy wielokrokowych gier obronnych Stackelberga.

Rozdział 7

Analiza algorytmu

W tym rozdziale zostanie przeprowadzona całościowa analiza zaproponowanego algorytmu ewolucyjnego - EASG. Zaprezentowane zostaną eksperymenty, na podstawie których dobierane były rekomendowane wartości parametrów, wpływ doboru parametrów na jakość wyników i wrażliwość algorytmu na ich zmianę. Również przedstawione zostaną dodatkowe modyfikacje metody oraz różne warianty operatorów. Analiza odbędzie się na podstawie 3 typów gier opisanych w rozdziale 6: Warehouse Games (WHG), Search Games (SEG) oraz FlipIt Games (FIG)

7.1 Wpływ parametrów na działanie algorytmu

W celu znalezienia najlepszych wartości parametrów algorytmu EASG przeprowadzone zostały eksperymenty na losowych 50 grach WHG, 20 grach SEG i 10 grach FIG. Należy podkreślić, że gry używane do wyznaczenia wielkości parametrów nie wchodziły w skład eksperymentów, na podstawie których zebrane zostały wyniki zaprezentowane w poprzednim rozdziale. Algorytm uruchomiony był 1000 razy dla każdej z gier z losowymi wartościami parametrów spośród podanych w tabeli 7.1. We wszystkich uruchomieniach każdy z parametrów losowany był niezależnie od pozostałych, a następnie wszystkie wyniki dla danej wartości parametru zostały uśrednione.

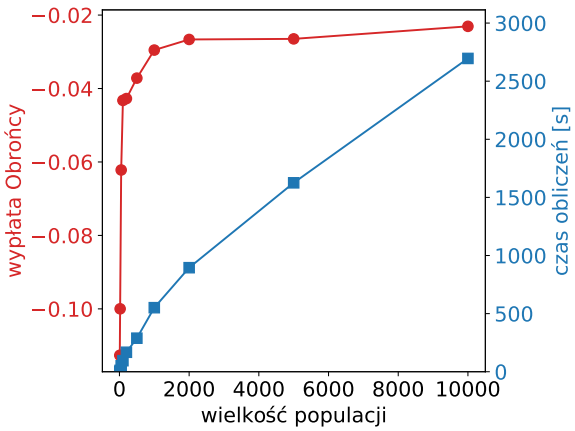
Rysunki 7.1 - 7.4 przedstawiają wyniki eksperymentów strojenia parametrów. Na wykresach zaprezentowano zależność uzyskanej wypłaty Obrońcy oraz czasu działania algorytmu od wielkości poszczególnych parametrów.

Parametr	**Symbol**	**Wartość**
Wielkość populacji	N	10, 20, 100, **200**, 500, 1000, 2000, 5000, 1000
Prawdopodobieństwo mutacji	p_m	0, 0.1, 0.2, 0.3, 0.4, **0.5**, 0.6, 0.7, 0.8, 0.9, 1
Prawdopodobieństwo krzyżowania	p_k	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, **0.8**, 0.9, 1
Presja selekcji	p_s	0.6, 0.7, 0.8, **0.9**, 0.95, 1
Wielkość elity	e	0, 1, **2**, 4, 8
Maksymalna liczba pokoleń	g_l	50, 100, **200**, 500, 1000
Maks. liczba pokoleń bez poprawy	g_k	5, 10, **20**, 50, 100

Tabela 7.1: Wartości parametrów branych pod uwagę w procesie strojenia algorytmu. Ostatecznie wybrane wartości zostały **pogrubione**.

Wielkość populacji

Z rysunku 7.1 można wywnioskować, że zwiększanie wielkości populacji korzystnie wpływa na otrzymywane wyniki. Jest to dość intuicyjna obserwacja, ponieważ większa liczba osobników przeszukujących przestrzeń, oznacza potencjalnie większą liczbę sprawdzonych strategii kandydatów i dokładniejsze przeszukanie tej przestrzeni. Wzrost otrzymywanych wypłat jest szczególnie widoczny dla małych wielkości populacji ($10 \leqslant N \leqslant 100$), potem trend ten zaczyna zanikać i różnice w wypłatach dla większej liczby osobników są niewielkie. Czas obliczeń skaluje się w przybliżeniu liniowo wraz ze wzrostem N. Widać więc, że od pewnego momentu ($N \geqslant 2000$) ponoszone koszty obliczeniowe związane z rozszerzaniem wielkości populacji są niewspółmierne do osiąganych efektów (wzrostu wypłaty). Rekomendowaną wielkość populacji ustalono na $N = 200$.



Rysunek 7.1: Średnia wypłata Obrońcy oraz czas obliczeń w zależności od wielkości populacji (N).

Prawdopodobieństwo mutacji

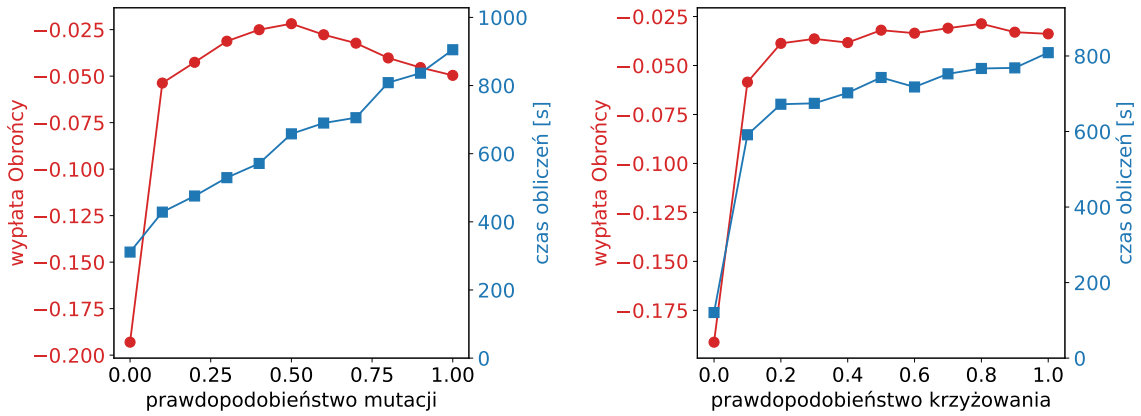
Relacja między wielkością wypłaty Obrońcy a prawdopodobieństwem mutacji zademonstrowana jest na rysunku 7.2a. Pierwszym wnioskiem, jaki można na jego podstawie wysnuć, jest istotność procesu mutacji. Bez niego ($p_m = 0$) następuje znaczny spadek wypłaty Obrońcy. Wynika to z faktu, że mutacja odpowiada za eksplorację przestrzeni przeszukiwań, tworzy nowe strategie. Bez niej, jedynie strategie proste wylosowane na początku są ze sobą mieszane w procesie krzyżowania. Z drugiej strony zbyt wysokie prawdopodobieństwo mutacji również powoduje obniżenie wyników. W takim przypadku ($p_m > 0.5$) większość strategii jest modyfikowana w każdym pokoleniu, co oznacza, że pojawia się dużo zaburzonych w sposób losowy strategii, które w większości nie mają wysokich wypłat i zaczynają dominować w populacji. Na zaprezentowanym wykresie widać maksimum wypłat osiągane dla $p_m = 0.5$ i dlatego też taka wartość została przyjęta w eksperymentach. Czas działania algorytmu rośnie nieznacznie wraz ze wzrostem prawdopodobieństwa mutacji, ponieważ większa liczba mutacji oznacza więcej operacji obliczeniowych do przeprowadzenia. Jednak wzrost ten nie jest tak znaczący jak w przypadku wcześniej rozważanego parametru - wielkości populacji.

Prawdopodobieństwo krzyżowania

Podobnie do mutacji, również krzyżowanie okazało się być kluczowym elementem efektywnego działania EASG. Rysunek 7.2b przedstawia zdecydowany spadek osiąganych wyników w przypadku rezygnacji z krzyżowania ($p_k = 0$). Jednocześnie różnice pomiędzy rezultatami dla pozostałych przetestowanych wartości prawdopodobieństwa krzyżowania nie są istotne. Naturalnie im wyższa wartość tego parametru, tym dłuższy czas obliczeń jednak również w tym przypadku wzrost nie jest znaczący. Na podstawie otrzymanych wyników arbitralnie przyjęto $p_k = 0.8$ jak rekomendowaną wartość prawdopodobieństwa krzyżowania.

Presja selekcji

Presja selekcji mniejsza niż 0.5 ($p_s < 0.5$) oznaczałaby większą szansę na promocję (zwycięstwo w turnieju) osobników o mniejszej wartości funkcji przystosowania, co naturalnie nie jest pożądaną własnością algorytmu ewolucyjnego. Z tego względu jedynie



(a) Prawdopodobieństwo mutacji (p_m). (b) Prawdopodobieństwo krzyżowania (p_k).

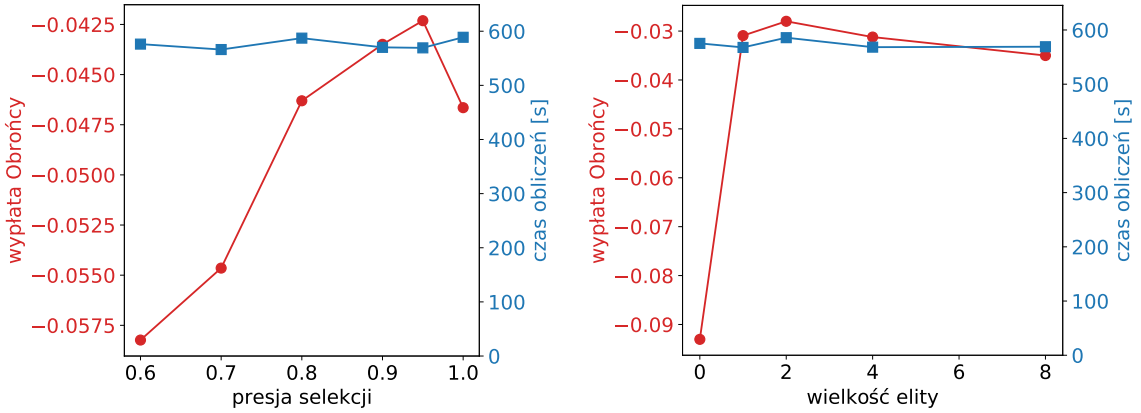
Rysunek 7.2: Średnia wypłata Obrońcy oraz czas obliczeń w zależności od prawdopodobieństwa mutacji (p_m) i krzyżowania (p_k).

wartości $p_s > 0.5$ zostały uwzględnione w eksperymentach. Rysunek 7.3a przedstawia wyniki testów tego parametru. Najlepsze rezultaty były osiągane dla $p_s = 0.9$. Zarówno mniejsze wartości (częstsza promocja słabiej przystosowanych osobników) jak i większe powodowały obniżenie jakości wyników, lecz różnice nie były duże. Spadek wypłat dla $p_s = 1$ (zawsze lepiej przystosowany osobnik jest promowany do następnego pokolenia) można tłumaczyć zbyt małą różnorodnością populacji. Przyglądając się dokładniej poszczególnym przebiegom, można zauważyć sytuacje, w których, dzięki promocji słabszego osobnika, w kolejnym pokoleniu poprzez mutację lub krzyżowanie powstaje z niego najlepiej przystosowany osobnik w całej populacji. Bez promocji mniej przystosowanych osobników algorytm czasami utyka w optimach lokalnych, a pojedyncze mutacje są niewystarczające, aby przenieść "uwagę" algorytmu w inny obszar poszukiwań. Presja selekcji w żaden sposób nie wpływa na liczbę obliczeń - to jedynie liczba, która decyduje, który z uczestników turnieju zostanie zwycięzcą i znajdzie się w kolejnym pokoleniu. Stąd stały czas obliczeń niezależnie od przyjętej wartości tego parametru.

Wielkość elity

Kolejnym rozważanym parametrem była wielkość elity, czyli liczba najlepiej przystosowanych osobników, które są bezwarunkowo promowane do następnego pokolenia. Wyniki zaprezentowane na rysunku 7.3b pokazują, że brak elitaryzmu ($e = 0$) znacznie

pogarsza otrzymywane wyniki. W takim przypadku istnieje szansa, że najlepsze znalezione rozwiązanie nie zostanie w procesie selekcji przeniesione do kolejnego pokolenia i zostanie przez algorytm "zapomniane". W przypadku pozostałych wartości ($e > 0$) różnice pomiędzy otrzymywanymi wypłatami są nieznaczące. Wydaje się jednak, że od pewnego momentu zbyt duży elitaryzm może prowadzić do zmniejszenia różnorodności populacji i tym samym do obniżenia wyników. Podobnie jak w przypadku presji selekcji, także wielkość elity nie wpływa na czas obliczeń, który jest w przybliżeniu stały niezależnie od przyjętej wartości. Ostatecznie wybrano $e = 2$ jako rekomendowaną wartość.



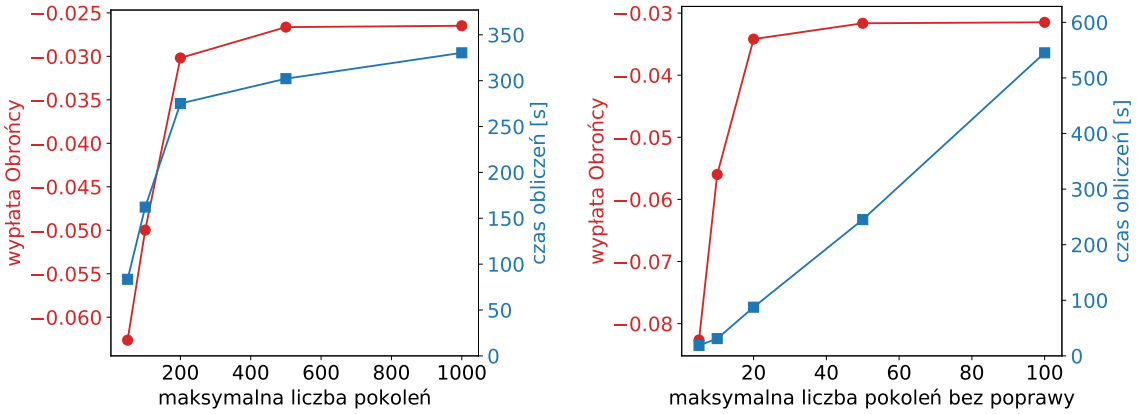
(a) Presja selekcji (p_s). (b) Wielkość elity (e).

Rysunek 7.3: Średnia wypłata Obrońcy oraz czas obliczeń w zależności od presji selekcji (p_s) i wielkości elity (e).

Warunek stopu

Ostatnie dwa testowane parametry związane są z warunkiem stopu algorytmu i są to: maksymalna liczba pokoleń (g_l) oraz maksymalna liczba pokoleń bez poprawy rozwiązania (g_k). Dla obu tych parametrów otrzymane rezultaty zaprezentowane na rysunku 7.4 mają podobną charakterystykę. Małe wartości ($g_l < 200$ i $g_k < 20$) skutkują niskimi wypłatami. Przyczyną jest najpewniej fakt, że w tych przypadkach zatrzymanie algorytmu następuje zbyt szybko, zanim zdąży on znaleźć dobre jakościowo rozwiązania. Podczas dokładniejszej analizy poszczególnych przebiegów algorytmu można zaobserwować przypadki, w których algorytm przez kilka lub kilkanaście pokoleń nie potrafi poprawić osiągniętego najlepszego wyniku, a następnie w wyniku wylosowania

odpowiedniej mutacji lub krzyżowania następuje znaczna poprawa. Niska wartość g_k oznacza, że w takich sytuacjach działanie algorytmu może zostać zakończone, zanim ta poprawa nastąpi. Na zademonstrowanych wykresach widać jednak, że wyższe wartości tych parametrów nie skutkują znacznym wzrostem wypłaty Obrońcy. Algorytm w pewnym momencie nie jest zdolny do poprawy znalezionego rozwiązania i nawet wydłużenie czasu jego działania (odroczenie warunku stopu) nie powoduje istotnego zysku w otrzymywanych rezultatach. Dlatego też jako kompromis między czasem obliczeń a jakością znajdowanych rozwiązań, jako rekomendowane wartości parametrów związanych z warunkiem stopu przyjęto $g_l = 200$ i $g_k = 20$.



(a) Maksymalna liczba pokoleń (g_l).

(b) Maksymalna liczba pokoleń bez poprawy (g_k).

Rysunek 7.4: Średnia wypłata Obrońcy oraz czas obliczeń w zależności od maksymalnej liczby pokoleń (g_l i g_k).

Powyższa analiza wpływu poszczególnych parametrów na uzyskiwane wyniki nie tylko pokazuje pewne własności zaproponowanego rozwiązania i wskazuje rekomendowane wartości, ale może też być wskazówką do indywidualnego dopasowania algorytmu do specyficznych potrzeb. Przykładowo, jeśli mniej istotny jest czas oczekiwania na zwrócony rezultat, a większy nacisk kładziony jest na jakość wyników, to poprzez zwiększenie wielkości populacji oraz maksymalnej liczby pokoleń można spodziewać się niewielkiej poprawy osiąganych rezultatów kosztem czasu działania. Praktycznym przykładem może tu być optymalizacja harmonogramu obchodu budynku przez pracowników ochrony. Jest to środowisko, które zmienia się rzadko, więc raz wyliczona strategia może służyć przez wiele miesięcy. W takim przypadku czas jej wyliczenia nie

jest istotny - akceptowalne są godziny lub nawet dni. Jednak nadal może okazać się, że wyliczenie strategii przy pomocy jednej z metod dokładnych jest niemożliwe ze względu na znaczne zasoby pamięciowe, jakich te metody wymagają lub skomplikowanie problemu (czas obliczeń rośnie wykładniczo).

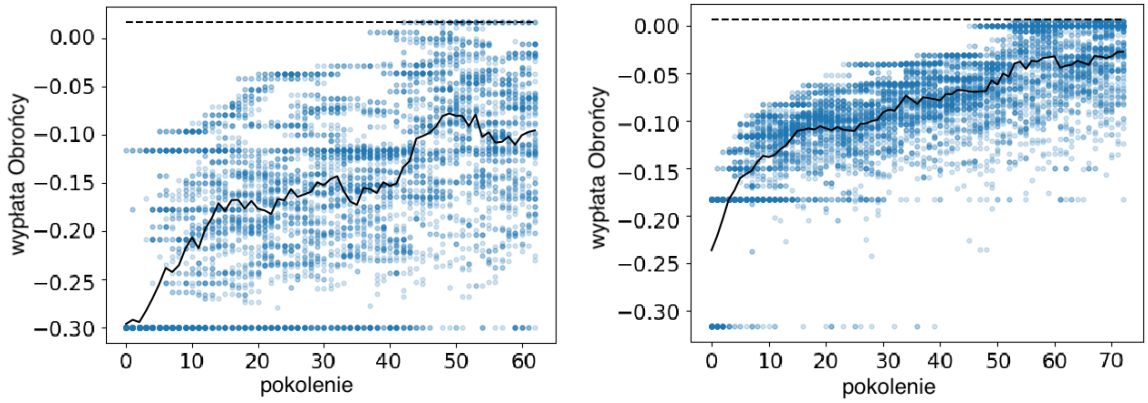
7.2 Zbieżność

Charakterystyczną cechą algorytmów ewolucyjnych jest potwierdzona skuteczność eksperymentalna przy jednoczesnym braku ścisłych dowodów matematycznych ich zbieżności czy teoretycznych gwarancji osiąganych rezultatów. Przyjętą w literaturze praktyką jest weryfikacja eksperymentalna proponowanych algorytmów ewolucyjnych poprzez szereg testów na zestawie zbiorów benchmarkowych. Najczęściej nie dokonuje się analizy teoretycznej takich metod, która jest zwykle zbyt skomplikowana lub niemożliwa do przeprowadzenia. Takie podejście jest prezentowane również w tej rozprawie. W rozdziale 6 zaprezentowano skuteczność zaproponowanego rozwiązania na zbiorze różnych gier testowych. W tym rozdziale przedstawiona zostanie typowa charakterystyka zbieżności proponowanego algorytmu oraz uzasadniona będzie jego teoretyczna zdolność osiągania optymalnego rozwiązania.

7.2.1 Charakterystyka zbieżności

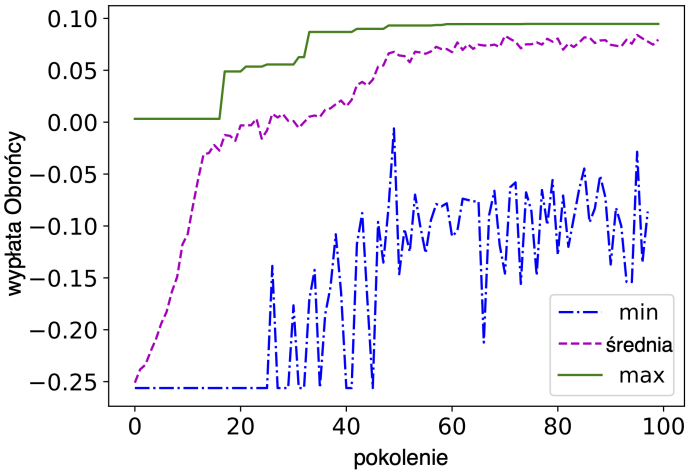
Rysunek 7.5 przedstawia wizualizację dwóch typowych przebiegów algorytmu EASG. Na ich podstawie można zaobserwować kilka cech charakterystycznych działania proponowanej metody, które są wspólne dla większości przeprowadzonych eksperymentów. W populacji początkowej wysokości wypłat skupione są głównie wokół jednej lub kilku wartości. Przypomnijmy, że populacja początkowa złożona jest jedynie ze strategii prostych. Podczas gry przeciwko takiej strategii Atakujący ma pewność akcji, które wykona Obrońca, więc wybiera najbardziej opłacalny niechroniony cel. Uzyskane wypłaty w tym przypadku nie zależą od prawdopodobieństw strategii wchodzących w skład strategii mieszanej Obrońcy (bo jest tylko jedna strategia prosta z prawdopodobieństwem 1), a od wypłat przypisanych do celów - stąd wiele strategii prostych skutkuje tą samą wypłatą.

W kolejnych pokoleniach wartości wypłat są bardziej zróżnicowane, ponieważ algorytm poprzez krzyżowania i mutacje tworzy bardziej różnorodne strategie mieszane. Co ważne, widoczny jest trend przesuwania się populacji ku wyższym wartościom wypłat przy jednoczesnym zachowaniu jej różnorodności. W dalszych pokoleniach nadal występują osobniki z niską wypłatą, ponieważ są one skutkiem działania mutacji, która w losowy sposób zmienia strategie i poszukuje nowych rozwiązań. Jednak widać, że średnia wartość przystosowania populacji (zaznaczona ciągłą linią) wykazuje trend wzrostowy w kolejnych pokoleniach. Ostatecznie pojawiają się osobniki, które osiągają optimum (linia przerywana) i po $g_k = 20$ kolejnych pokoleniach bez poprawy wyniku aktywowany jest warunek stopu, który kończy działanie algorytmu.



Rysunek 7.5: Wizualizacja typowych przebiegów algorytmu EASG. Każdy z punktów oznacza wartość funkcji przystosowania (wypłaty Obrońcy) pojedynczego osobnika w kolejnych pokoleniach. Przerywana linia to optymalny wynik, linią ciągłą oznaczono średnią wypłatę osobników z danego pokolenia.

Powyższe spostrzeżenia potwierdza rysunek 7.6, na którym zaprezentowano typową charakterystykę działania algorytmu. Wypłata reprezentowana przez najsłabszego osobnika zmienia się dość dynamicznie - zdarza się, że mutacja tworzy osobniki słabsze, więc czasami wartość ta maleje. Jednak średnie przystosowanie osobników posiada trend wzrostowy (choć w sposób naturalny i tu czasami pojawiają się spadki między kolejnymi pokoleniami). Wartość przystosowania najlepszego osobnika nigdy nie maleje dzięki zastosowaniu procedury elitaryzmu, która przechowuje najlepsze osobniki. Czasami wzrost przystosowania najlepszego osobnika rośnie skokowo, ponieważ w efekcie mutacji lub krzyżowania może powstać strategia znacznie lepsza niż dotychczas znalezione.



Rysunek 7.6: Przykładowa reprezentatywna charakterystyka zmiany wypłaty w kolejnych pokoleniach.

7.2.2 Zdolność do generowania dowolnej strategii

Przedstawione w rozdziale 5 operatory ewolucyjne mutacji i krzyżowania zostały zaprojektowane w sposób umożliwiający wygenerowanie przy ich pomocy dowolnej strategii mieszanej Obrońcy. Aby to uzasadnić, poczyńmy 3 następujące spostrzeżenia:

A. Mutacja może wygenerować dowolnie wybraną strategię prostą.

Uzasadnienie: Operator mutacji najpierw losuje krok czasowy, począwszy od którego przypisuje w sposób losowy kolejne akcje, wybierając spośród wszystkich możliwych. Zatem w przypadku wylosowania kroku pierwszego zmieniana jest cała strategia prosta, a w wyniku tej zmiany może powstać dowolna inna strategia.

B. Krzyżowanie może stworzyć strategię mieszaną złożoną z dowolnie wybranych strategii prostych.

Uzasadnienie: Krzyżowanie łączy ze sobą zbiory złożone ze strategii prostych - w rezultacie powstaje suma zbiorów. W oczywisty sposób takie łączenia mogą wygenerować dowolny zbiór strategii prostych, o ile będą one dostępne w puli wszystkich strategii prostych w chromosomach całej populacji (jak pokazano w punkcie A., dzięki mutacji mogą zostać stworzone dowolne strategie proste).

C. Krzyżowanie może przypisać danej strategii prostej dowolnie wybrane prawdopodobieństwo.

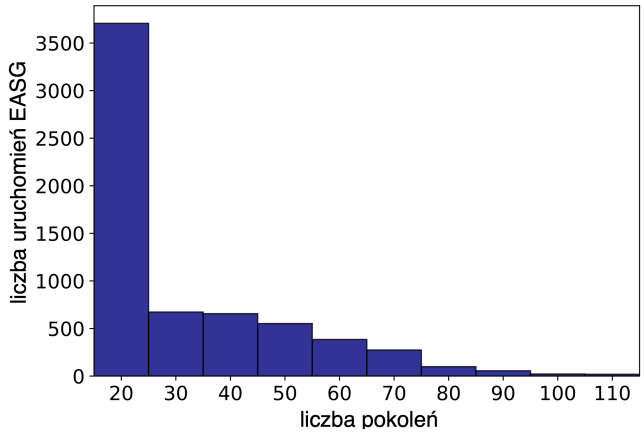
Uzasadnienie: Krzyżowanie łącząc dwa zbiory strategii prostych i przepoławia ich prawdopodobieństwa. Zatem kolejne krzyżowania generują prawdopodobieństwa będące potęgami $\frac{1}{2}$. Dodatkowo połączenie zbiorów z identyczną strategią prostą powoduje dodanie odpowiadających im prawdopodobieństw. Jak wiadomo każdą liczbę z przedziału $[0, 1]$ można dowolnie przybliżyć sumą odwrotności potęg 2 - analogicznie jak zapisuje się część ułamkową liczby w systemie dwójkowym, więc na podobnej zasadzie każdą wartość prawdopodobieństwa można uzyskać poprzez wielokrotne zastosowanie odpowiednich krzyżowań.

Łącząc wszystkie trzy powyższe spostrzeżenia, można stwierdzić, że przy pomocy mutacji i krzyżowania możliwe jest wygenerowanie dowolnej strategii mieszanej. Oczywiście w powyższych rozważaniach abstrahujemy od prawdopodobieństwa takiego zdarzenia, które będzie zależało od rodzaju gry, jej wielkości oraz postaci generowanej strategii. Pokazano jedynie teoretyczną możliwość tworzenia przez EASG dowolnych strategii, a więc również tej optymalnej. Co więcej, można pokazać, że możliwe jest też przejście za pomocą skończonej liczby operacji mutacji i krzyżowania pomiędzy dwoma dowolnymi strategiami mieszanymi.

7.3 Analiza działania algorytmu

7.3.1 Szybkość zbieżności

Liczba pokoleń generowanych przez algorytm do momentu spełnienia warunku stopu (czyli szybkość zbieżności) silnie zależy od danego problemu i jego trudności. W niektórych przypadkach optymalna strategia była znajdowana bardzo szybko - nawet już w drugim pokoleniu. Z drugiej strony zdarzały się instancje gier, dla których warunek stopu był spełniony dopiero po ponad 100 generacjach. Średnia liczba pokoleń we wszystkich eksperymentach wynosiła 34. Rysunek 7.7 przedstawia częstotliwość liczby pokoleń wygenerowanych przez algorytm. W większości przypadków zwrócona strategia była znajdowana stosunkowo szybko - w pierwszych 25 pokoleniach.

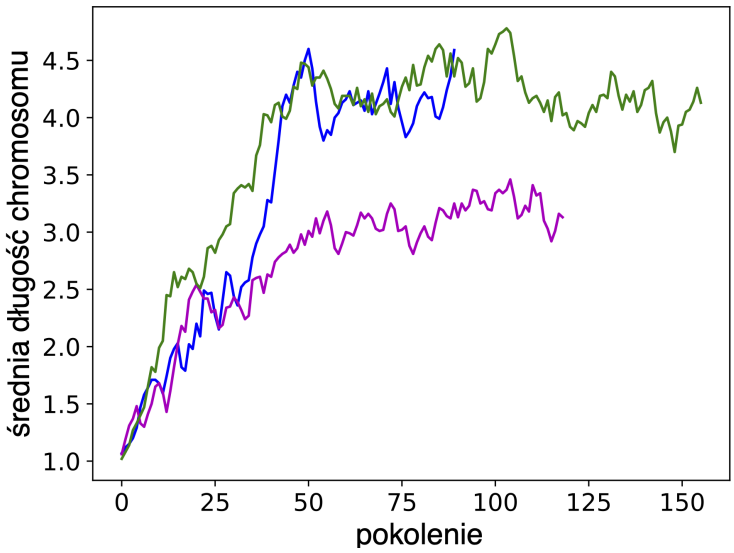


Rysunek 7.7: Histogram liczby pokoleń wygenerowanych przez algorytm we wszystkich eksperymentach.

7.3.2 Wielkość chromosomów

Aby dokładniej zrozumieć działanie zaproponowanego algorytmu, warto przyjrzeć się długościom chromosomów, czyli liczbom strategii prostych, które wchodzą w skład zakodowanych w osobnikach strategii mieszanych. Rysunek 7.8 prezentuje średnią wartość tego parametru dla 3 przykładowych reprezentatywnych przebiegów algorytmu. W trakcie początkowych pokoleń średnia długość chromosomów rośnie w sposób zbliżony do liniowego. W okolicy 50 pokolenia wartość ta stabilizuje się i osiąga wartość w zakresie od około 3 do 5. Stabilizacja związana jest z charakterystyką operatora krzyżowania - szansa usunięcia danej strategii prostej jest większa, jeśli jej prawdopodobieństwo jest małe, a im więcej strategii prostych w chromosomie, tym mniejsze jest ich średnie prawdopodobieństwo. Krzyżowanie więc niejako blokuje powstawanie chromosomów o znacznej długości.

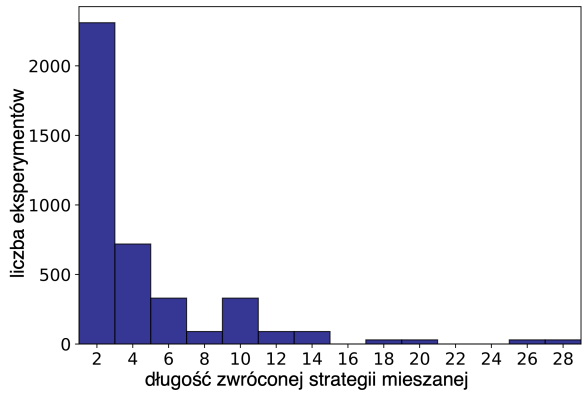
Liczba strategii prostych w optymalnym (wyznaczonym algorytmem dokładnym) rozwiązaniu silnie zależy od konkretnej instancji gry i waha się między 2 a 28. Strategie zwracane przez metodę EASG w większości mają rozmiar między 2 a 5, znacznie rzadziej są to strategie mieszane posiadające ponad 10 strategii prostych. Rozkład tych wartości dla wszystkich uruchomień 3 typów gier przedstawiony jest w postaci histogramów na rysunku 7.9.



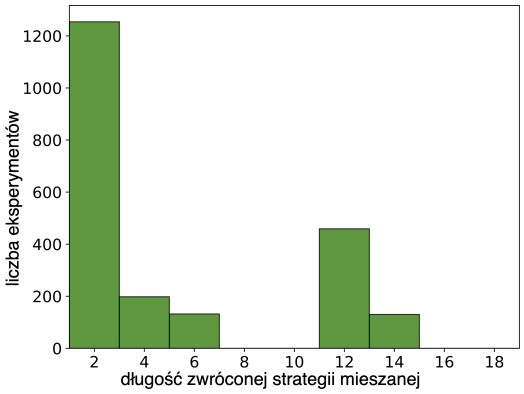
Rysunek 7.8: Średnia długość chromosomów (liczba strategii prostych) podczas 3 przykładowych przebiegów algorytmu EASG.

7.3.3 Wskaźnik sukcesu mutacji

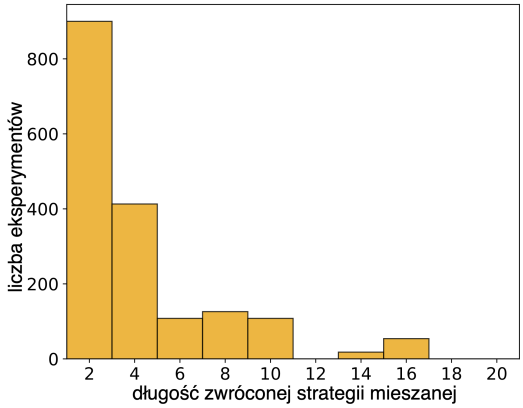
Kolejnym ważnym parametrem działania algorytmu jest wskaźnik sukcesu mutacji, czyli odsetek operacji mutacji, dla których strategia zmodyfikowana przez tę operację miała wyższą wartość funkcji przystosowania niż przed modyfikacją. Histogramy prezentujące rozkład tego wskaźnika dla różnych typów gier zaprezentowane są na rysunku 7.10. Dla gier WHG mutacja poprawia wynik dość często - około 30-35% mutacji kończy się sukcesem w postaci lepiej przystosowanego osobnika. W pozostałych dwóch grach ten współczynnik jest dużo niższy. To zjawisko można wyjaśnić charakterystyką zasad tych gier oraz wielkością przestrzeni przeszukiwań. Przykładowo w grach typu SEG większość strategii składowych odpowiada przypadkowi odkrycia śladów obecności Atakującego (patrz rozdział 6.2). Jednak gdy strategia Atakującego polega na zacieraniu wszystkich śladów, to zmiana tych strategii Obrońcy (aktywujących się w sytuacji wykrycia śladów) nie będzie miała żadnego wpływu na oczekiwaną wypłatę, bo znaczenie będzie miała jedynie ta strategia zakładająca brak wykrytych śladów. W efekcie zaaplikowana mutacja do większości strategii nie zmieni wartości przystosowania osobnika, co obniża odsetek sukcesów mutacji w tym typie gier.



(a) Warehouse Games.



(b) Search Games.

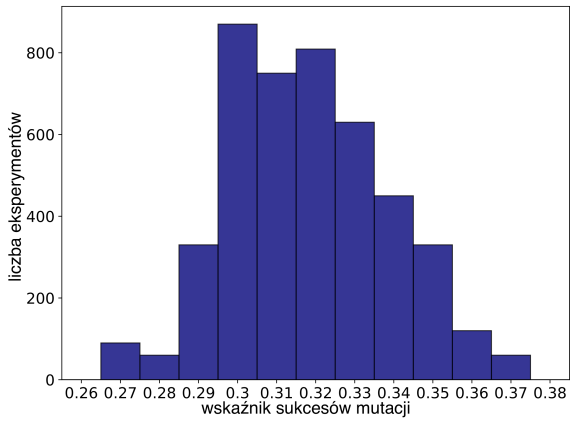


(c) FlipIt Games.

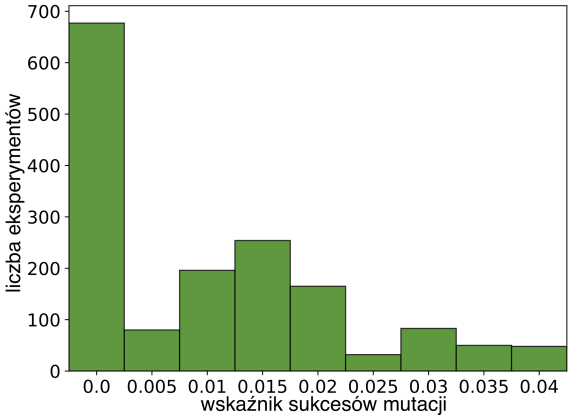
Rysunek 7.9: Histogramy przedstawiające liczbę strategii prostych wchodzących w skład zwracanych rozwiązań z wszystkich przeprowadzonych eksperymentów.

7.3.4 Stabilność

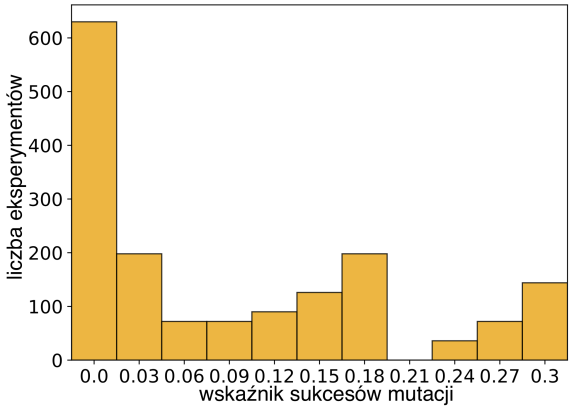
Metoda EASG jest silnie niedeterministyczna - losowana jest populacja początkowa, operatory mutacji i krzyżowania również zawierają elementy losowości, podobnie selekcja. W przypadku takich metod sama możliwość osiągania optymalnych wyników (zaprezentowana w rozdziale 6) nie jest jedynym i wystarczającym kryterium jakości metody. Równie istotnym aspektem jest stabilność, czyli zdolność algorytmu do powtarzalnego zwracania wyników. Jedną z podstawowych metryk, dzięki której można zweryfikować stabilność danego rozwiązania, jest odchylenie standardowe wyników policzone na podstawie wielokrotnych uruchomień algorytmu dla tej samej instancji problemu. W ten sposób zbadana została stabilność metody EASG. Rysunek 7.11 pokazuje rozkłady odchyleń standardowych z 20 uruchomień policzonych dla wszystkich 300 gier biorących udział w eksperymentach. W 145 przypadkach (48%) odchylenie



(a) Warehouse Games.



(b) Search Games.



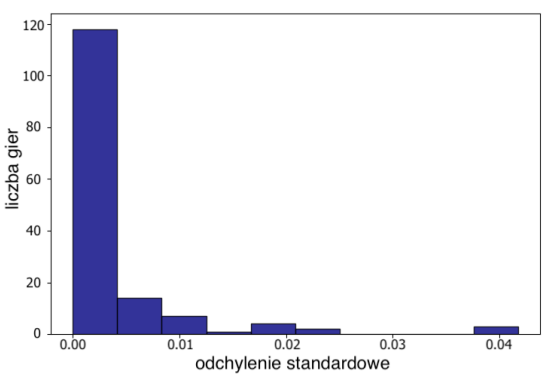
(c) FlipIt Games.

Rysunek 7.10: Rozkład wskaźnika sukcesów mutacji (odsetka mutacji, które spowodowały poprawę rozwiązania) dla wszystkich przeprowadzonych eksperymentów.

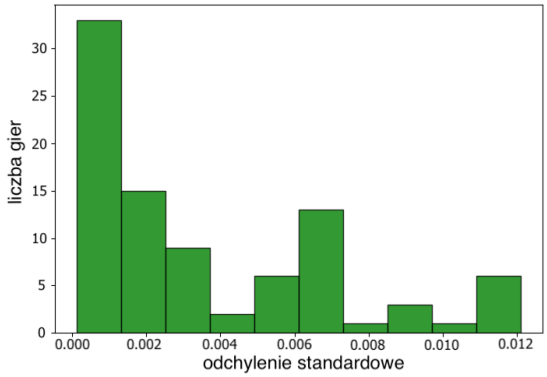
standardowe było równe 0, czyli wszystkie 20 uruchomień zwróciło ten sam wynik. W tym zbiorze znalazło się 96 gier WHG, 31 gier SEG i 18 gier FIG. Średnie odchylenie standardowe wyniosło 0.0059 z maksymalną wartością 0.1629, co stanowi 36.7% zakresu wypłat (różnicy między maksymalną i minimalną możliwą wypłatą).

Dla 133 gier, w co najmniej jednym z 20 uruchomień metoda EASG zwróciła optymalne rozwiązanie. W 82 (62%) przypadkach spośród nich optymalne rozwiązanie zwrócone było we wszystkich 20 uruchomieniach. Dla 96 z tych gier (72%) optymalne rozwiązanie było znajdowane w więcej niż 90% uruchomień. Z drugiej strony w 13 przypadkach (9%) optymalna strategia została zwrócona tylko jednokrotnie.

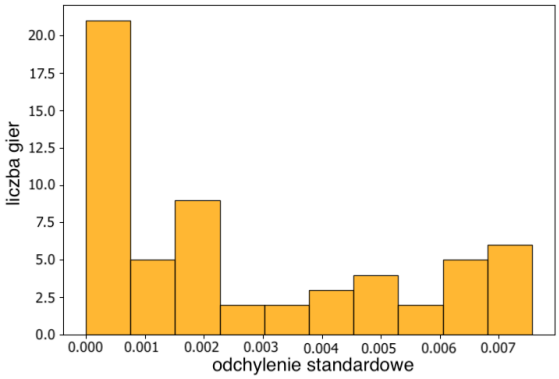
Na podstawie wyżej zaprezentowanych wyników można uznać, że zaproponowana metoda jest stabilna i potrafi w sposób powtarzalny zwracać wyniki optymalne lub bliskie optymalnym.



(a) Warehouse Games.



(b) Search Games.



(c) FlipIt Games.

Rysunek 7.11: Rozkład odchyleń standardowych dla przetestowanych gier.

7.4 Dodatkowe modyfikacje

W tej części zostaną przedstawione wyniki dodatkowych 14 modyfikacji wybranych elementów omawianego algorytmu. Zostały one przetestowane na tym samym zestawie gier WHG, SEG i FIG co oryginalne rozwiązanie.

Warianty EASG:

- **EASG$_{\mathbf{MNSP}}$** - *mutacja z nowymi strategiami prostymi* - operator mutacji jest rozszerzony o nową akcję: dodanie do chromosomu losowo wygenerowanej strategii prostej z przypisanym losowym prawdopodobieństwem* (po tej operacji wszystkie prawdopodobieństwa są normalizowane, aby ich suma wynosiła 1).

- **EASG$_{\mathbf{MNP}}$** - *mutacja z nowym prawdopodobieństwem* - operator mutacji jest rozszerzony o nową akcję: losowo wybranej strategii prostej zostaje przypisana nowa, wylosowana z rozkładu jednostajnego, wartość prawdopodobieństwa* (po

tej operacji wszystkie prawdopodobieństwa są normalizowane, aby ich suma wynosiła 1).

- **EASG$_{\mathbf{MZP}}$** - *mutacja z zamianą prawdopodobieństw* - operator mutacji zostaje rozszerzony o nową akcję: prawdopodobieństwa dwóch losowo wybranych strategii prostych zostają zamienione miejscami*.

- **EASG$_{\mathbf{MUSP}}$** - *mutacja z usunięciem strategii prostej* - operator mutacji jest rozszerzony o nową akcję: losowo wybrana strategia prosta jest usuwana z chromosomu* (jeśli nie jest jedyną strategią w chromosomie). Po tej operacji wszystkie prawdopodobieństwa są normalizowane, aby ich suma wynosiła 1.

- **EASG$_{\mathbf{MNSP_BP}}$** - *wariant $EASG_{MNSP}$ bez powtórzeń* - mutacja rozszerzona o dodanie nowej strategii (jak w wariancie EASG$_{\mathrm{MNSP}}$), ale zmodyfikowany osobnik jest zachowywany niezależnie od tego, czy jest on lepiej przystosowany.

- **EASG$_{\mathbf{MNP_BP}}$** - *wariant $EASG_{MNP}$ bez powtórzeń* - mutacja rozszerzona o losową modyfikację prawdopodobieństwa jednej strategii prostej (jak w wariancie EASG$_{\mathrm{MNP}}$), ale operacja ta wykonywana jest jednokrotnie niezależnie od tego, czy zmieniony osobnik jest lepiej przystosowany.

- **EASG$_{\mathbf{MZP_BP}}$** - *wariant $EASG_{MZP}$ bez powtórzeń* - mutacja rozszerzona o zamianę prawdopodobieństw losowo wybranych strategii prostych (jak w wariancie EASG$_{\mathrm{MZP}}$), ale operacja ta wykonywana jest jednokrotnie niezależnie od tego, czy zmieniony osobnik jest lepiej przystosowany.

- **EASG$_{\mathbf{MUSP_BP}}$** - *wariant $EASG_{MUSP}$ bez powtórzeń* - mutacja jest rozszerzona o usuwanie z chromosomu losowej strategii prostej (jak w wariancie EASG$_{\mathrm{MUSP}}$), lecz operacja ta zostaje wykonana jednokrotnie niezależnie od kierunku zmiany przystosowania modyfikowanego osobnika.

*Jeśli po zastosowaniu mutacji otrzymany osobnik ma niższą wartość funkcji przystosowania, to chromosom jest przywracany do stanu sprzed zastosowania mutacji. Mutacja jest powtarzana, dopóki nie zostanie wygenerowany lepiej przystosowany osobnik lub osiągnięty będzie limit m_{limit} prób. W eksperymentach przyjęto $m_{limit} = 50$.

- **EASG$_{\mathbf{MUNS}}$** - *mutacja z usuwaniem najsłabszej strategii prostej* - operator mutacji jest rozszerzony o nową akcję: usunięcie strategii prostej, której wypłata jest najniższa**. Po wykonaniu tej operacji pozostałe prawdopodobieństwa w chromosomie zostają znormalizowane, aby ich suma wynosiła 1.

- **EASG$_{\mathbf{KUW}}$** - *krzyżowanie uwzględniające wypłaty* - operator krzyżowania zamiast usuwać strategie proste bazując na ich prawdopodobieństwach (szansa na usunięcie strategii jest tym większa, im niższe jest jej prawdopodobieństwo), oparty jest o wypłaty - prawdopodobieństwo usunięcia danej strategii prostej jest odwrotnie proporcjonalne do jej wypłaty**.

- **EASG$_{\mathbf{ZSP}}$** - *zachłanne strategie początkowe* - zamiast przypisywać do populacji początkowej zupełnie losowe strategie proste, najpierw generowane są wszystkie możliwe strategie proste, a następnie spośród nich wybieranych jest N z najwyższą wartością funkcji przystosowania, które tworzą populację początkową.

- **EASG$_{\mathbf{MNS}}$** - *mutacja najsłabszej strategii* - mutacja jest aplikowana zawsze do strategii prostej z najniższą wypłatą** zamiast do losowo wybranej strategii prostej.

- **EASG$_{\mathbf{MPW}}$** - *mutacja proporcjonalnie do wypłaty* - operator mutacji jest aplikowany do losowo wybranej strategii prostej, ale losowanie to odbywa się zgodnie z rozkładem zadanym przez wypłaty poszczególnych strategii prostych**, tj. im strategia ma wyższą wypłatę, tym mniejsza jest szansa na jej modyfikację w wyniku mutacji.

- **EASG$_{\mathbf{NO}}$** - *nowe osobniki* - w każdym pokoleniu dodawanych jest $0.05N$ nowych osobników do populacji. Osobniki te są generowane tak, jak jest to realizowane w populacji początkowej.

Tabela 7.2 prezentuje wyniki wszystkich wyżej opisanych wariantów algorytmu. Przedstawione zostały średnie wypłaty Obrońcy w podziale na 3 typy gier. Statystycz-

**Wypłata strategii prostej z chromosomu liczone jest przy założeniu, że Obrońca przyjmuje tę strategię jako swoją strategię w rozgrywce przeciwko Atakującemu.

na istotność różnicy między wariantem bazowym (EASG) a poszczególnymi modyfikacjami była policzona testem Wilcoxona z progiem istotności $p - \text{value} < 0.05$.

	średnia wypłata Obrońcy			**czas obliczeń** [s]		
wariant	**WHG**	**SEG**	**FIG**	**WHG**	**SEG**	**FIG**
EASG	**0.017**	0.108	0.031	152	2534	328
$\text{EASG}_{\text{MNSP}}$	0.016	**0.139**	0.036	1366	21892	2988
EASG_{MNP}	0.016	0.131	**0.037**	1285	22651	3008
EASG_{MZP}	0.016	0.108	**0.037**	1332	21447	2931
$\text{EASG}_{\text{MUSP}}$	0.013	0.053	0.026	1283	22026	2900
$\text{EASG}_{\text{MNSP\_BP}}$	0.014	0.059	0.031	156	2548	313
$\text{EASG}_{\text{MNP\_BP}}$	0.015	0.074	0.030	148	2422	336
$\text{EASG}_{\text{MZP\_BP}}$	0.013	0.099	0.024	156	2583	316
$\text{EASG}_{\text{MUSP\_BP}}$	0.013	0.052	0.029	147	2620	313
$\text{EASG}_{\text{MUNS}}$	0.008	0.058	0.018	139	2361	299
EASG_{KUW}	0.014	0.050	0.034	235	4013	516
EASG_{ZSP}	0.015	0.105	0.032	167	2892	365
EASG_{MNS}	0.013	0.046	0.031	148	2612	321
EASG_{MPW}	0.015	0.094	0.031	157	2642	336
EASG_{NO}	0.015	0.099	0.030	165	2735	328

Tabela 7.2: Średnia wartość wypłaty Obrońcy oraz czas obliczeń poszczególnych wariantów algorytmu. Najlepsze wyniki zostały **pogrubione**. Wyniki lepsze niż bazowa wersja algorytmu (EASG) zostały podkreślone. W przypadku gdy różnica wyników między wersją podstawową (EASG) a danym wariantem była statystycznie istotna, wynik został wyróżniony szarym tłem (zarówno w przypadku przewagi EASG jak i modyfikacji).

Kilka z przetestowanych wariantów zasługuje na szczególną uwagę. Są to głównie nowe warianty mutacji, których aplikacja jest powtarzana kilkukrotnie, aż do poprawy przystosowania mutowanego osobnika lub osiągnięcia maksymalnego limitu mutacji. Są to warianty $\text{EASG}_{\text{MNSP}}$, EASG_{MNP} oraz EASG_{MZP}. Poprawa wyników względem bazowej wersji algorytmu (EASG) widoczna jest szczególnie w grach SEG i FIG. Ma to związek z wcześniej poczynioną obserwacją dotyczącą wskaźnika sukcesów mutacji (patrz rysunek 7.10), który dla tych gier w przypadku EASG jest na dość niskim poziomie $(1 - 3\%)$. Dzięki kilkukrotnemu losowaniu mutacji, jak wynika z eksperymentów, częstotliwość sukcesów mutacji dla tych typów gier wzrasta nawet 8-10 razy.

W wypadku gier WHG poprawa nie jest widoczna - mutacja bez powtórzeń miała dość wysoki współczynnik sukcesów, więc dodanie powtórzeń (które zostają przery-

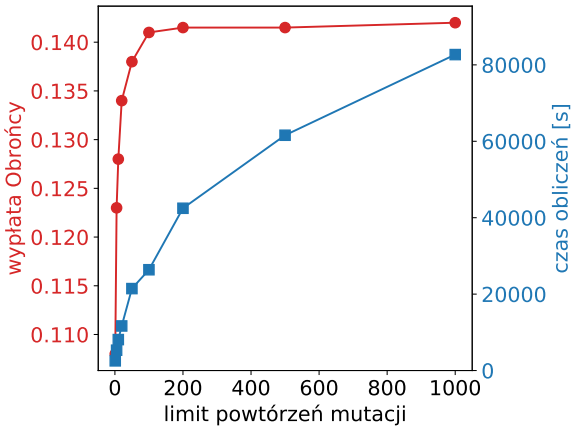
wane w przypadku znalezienia pierwszego lepszego rozwiązania) nie zmienia w sposób znaczący działania algorytmu. Należy jednak podkreślić, że takie "zachłanne" podejście do mutacji i każdorazowo próba zwiększenia wartości funkcji przystosowania ma dwie wady.

Po pierwsze czas działania algorytmu zostaje istotnie zwiększony, bowiem wielokrotna aplikacja mutacji generuje dodatkowe obliczenia - algorytm w każdym pokoleniu dla każdego mutowanego osobnika musi wielokrotnie wyliczać funkcję przystosowania po zaaplikowaniu każdej z mutacji (aby zdecydować czy potrzebne są kolejne jej powtórzenia). Jeśli wiele mutacji jest nieudanych (a zdarza się tak w przypadku osobników w optimum lokalnym, dla których niemożliwa jest poprawa wyniku przy pomocy pojedynczej operacji mutacji), to ewaluacja nowego rozwiązania (po mutacji) wykonywana jest wielokrotnie. Przypomnijmy, że ewaluacja wymaga sprawdzenia wszystkich strategii prostych Atakującego i jest czasochłonna. W efekcie poprawa wyników związana jest ze znacznym pogorszeniem czasu działania algorytmu, co potwierdzone jest w wynikach zaprezentowanych w tabeli 7.2.

Drugą słabością dążenia to poprawy rozwiązania w każdej mutacji jest zmniejszenie różnorodności populacji. W takim podejściu w populacji pojawiają się te same lub podobne osobniki, które są zbliżone do optimum lokalnego. Jednak nie zawsze jest to pożądane - czasami połączenie (w procesie krzyżowania) dwóch słabszych osobników lub kilkukrotne wykonanie mutacji (bez cofania jej skutków) może przynieść rozwiązanie lepsze niż najlepsza z pojedynczych mutacji, która "zamyka" algorytmowi drogę do tego rozwiązania.

W eksperymentach przyjęto limit powtórzeń mutacji $m_{limit} = 50$. Zasadne jest pytanie, jak prezentowałyby się wyniki dla innych wartości tego parametru i jak wpływa on na czas obliczeń. Te zależności dla wariantu $\text{EASG}_{\text{MNSP}}$ i gier SEG zostały przedstawione na rysunku 7.12. Dla początkowych wartości ($m_{limit} < 50$) wzrost wypłaty Obrońcy jest stosunkowo szybki. Dalsze zwiększanie tego parametru nie powoduje znacznego wzrostu otrzymywanych wyników. Zgodnie z wcześniejszym spostrzeżeniem powtarzanie mutacji jest procesem kosztownym obliczeniowo, co można zaobserwować na zaprezentowanym wykresie.

Dla identycznych wariantów jak wyżej omówione, ale bez wielokrotnego powtarza-



Rysunek 7.12: Średnia wypłata Obrońcy oraz czas obliczeń w zależności od limitu powtórzeń mutacji (m_{limit}) wariantu $\text{EASG}_{\text{MNSP}}$ dla gier Search Games.

nia mutacji, zarówno czas działania jak i osiągane wyniki są na podobnym poziomie jak w bazowej wersji algorytmu EASG. Przeprowadzone testy statystyczne nie wykazały istotnej różnicy.

W przypadku mutacji, która usuwa strategię prostą ($\text{EASG}_{\text{MUSP}}$ i $\text{EASG}_{\text{MUSP\_BP}}$) wyniki uległy pogorszeniu. Jak zostało wcześniej pokazane na rysunku 7.9 w przeważającej większości zwracane strategie składały się z mniej niż 5 strategii prostych. Można więc podejrzewać, że przy tak małej liczbie każda z nich odgrywa znaczącą rolę, więc usunięcie dowolnej z nich negatywnie wpływa na wynik.

Na podstawie zaprezentowanych wyników widać również, że wszystkie modyfikacje związane z większą częstotliwością zmiany słabych strategii prostych (warianty EASG_{MNS} - mutacja najsłabszej strategii, EASG_{MPW} - wybór strategii do mutacji proporcjonalnie do wypłaty) lub większym prawdopodobieństwem ich usuwania w krzyżowaniu (EASG_{KUW}) nie przyniosły poprawy wyników. Otrzymane wyniki są na porównywalnym lub słabszym poziomie niż EASG. Wynika to najprawdopodobniej z faktu, że rozpatrywanie jakości pojedynczych strategii prostych w oderwaniu od pozostałych elementów danej strategii mieszanej może nie być właściwym podejściem - nawet jeśli indywidualnie strategia prosta jest słaba, to często pełni ona kluczową rolę w całości, bowiem sprawia, że pewna niekorzystna z perspektywy Obrońcy decyzja jest dla Atakującego nieopłacalna.

Wariant algorytmu, w którym w populacji początkowej znajdują się najlepsze stra-

tegie proste Obrońcy ($EASG_{ZSP}$) nie spowodował poprawy wyników. Uzasadnienie jest tu podobne jak w poprzednim akapicie - nie zawsze optymalna strategia mieszana składa się ze strategii prostych, które dają dobre wypłaty w przypadku ich indywidualnego rozpatrywania.

Również ostatnia przetestowana modyfikacja związana z dodawaniem nowych osobników w trakcie kolejnych pokoleń ($EASG_{NO}$) nie spowodowała zwiększenia zwracanej wypłaty Obrońcy i można uznać, że jest to operacja neutralna z punktu widzenia jakości uzyskiwanych rezultatów.

Podsumowując, większość z przetestowanych wariantów nie osiągnęła statystycznie istotnie lepszych wyników niż bazowa metoda EASG. Tylko niektóre z zaproponowanych modyfikacji skutkowały poprawą rezultatów. We wszystkich przypadkach wiązało się to ze znacznym (około dziewięciokrotnym) wzrostem czasu obliczeń. Zatem w sytuacjach, gdzie mniejsze znaczenie ma koszt obliczeniowy, a kluczowe jest uzyskanie jak najlepszego wyniku, niektóre z zaproponowanych modyfikacji mogą stanowić użyteczną alternatywę.

Rozdział 8

Ograniczona racjonalność

Jedną z zasad równowagi Stackelberga i dotychczas prezentowanych rozwiązań jest domniemanie pełnej racjonalności graczy czyli założenie, że obaj gracze wybierają optymalne dla siebie strategie, nie mylą się, zawsze potrafią podjąć najlepszą decyzję, posiadają odpowiednie umiejętności i zasoby do znajdowania optymalnych rozwiązań. Jednak wiele gier obronnych Stackelberga zostało sformułowanych w odpowiedzi na praktyczne potrzeby zamodelowania sytuacji związanych z bezpieczeństwem. Bardzo często nie są to tylko teoretyczne rozważania hipotetycznych sytuacji, a modele realnych scenariuszy, które następnie zostają wprowadzane w życie i w bardzo konkretny sposób wpływają na poprawę bezpieczeństwa (przykłady takich wdrożeń przytoczone zostały w rozdziale 3.4). Takie systemy projektowane są najczęściej jako pomoc w wyborze optymalnej strategii Obrońcy, którym są na przykład firmy ochroniarskie czy inne instytucje zapewniające bezpieczeństwo. Ich "przeciwnikami" są złodzieje, terroryści, przestępcy. Są to osoby, których decyzje mogą być obarczone różnymi błędami, a na podejmowane przez nich działania może wpłynąć wiele czynników, które spowodują, że nie zawsze będą one optymalne. Zatem szczególnie w takiej sytuacji ważne jest, aby ta "niedoskonałość" potencjalnych przeciwników była uwzględniona przy projektowaniu systemów bezpieczeństwa. W teorii gier modelowanie takich scenariuszy nazywane jest *ograniczoną racjonalnością*.

Termin *ograniczona racjonalność* (ang. *bounded rationality*) został użyty po raz pierwszy w 1957 roku przez laureata nagrody Nobla w dziedzinie ekonomii Herberta Simona w książce zatytułowanej "*Models of Man*" [69]. Początkowo temat ten nie

cieszył się dużą popularnością wśród badaczy. Duży wzrost zainteresowania można zauważyć w latach 90'. Wynikał on z faktu, że wraz z rozwojem technologii i popularyzacją komputerów, coraz częściej modele teorii gier zaczęły być wykorzystywane w praktycznych zastosowaniach, a co za tym idzie, pojawiła się potrzeba modeli, które jak najlepiej odzwierciedlają rzeczywistość i uwzględniają między innymi aspekt niedoskonałości decyzji graczy.

Należy podkreślić, że niepełna racjonalność nie oznacza nieracjonalności. Pojęcie nieracjonalności czy braku racjonalności odnosi się do sytuacji, w której działania gracza są nieprzewidywalne, losowe, nielogiczne. Natomiast ograniczona racjonalność zakłada, że gracz dąży do wyboru strategii optymalnej, ale z pewnych względów nie potrafi tego osiągnąć. Wśród przyczyn takiego zachowania wymienia się głównie ograniczone możliwości kognitywne, czyli na przykład pewien limit wariantów, jakie gracz jest w stanie dokładnie przeanalizować, brak umiejętności dokładnej oceny danej sytuacji (wyliczenia oczekiwanej wypłaty), ograniczony czas na podjęcie decyzji, niejasno sprecyzowany cel, itp. Wiele prac z dziedziny psychologii wskazuje, że ludzie mają skłonność do upraszczania rzeczywistości, wybierania łatwiejszych ścieżek [19]. Wykazują one, że oszczędzanie umysłu to nie lenistwo, ale ochrona przed przeciążeniem systemu. Wraz ze wzrostem złożoności otaczającego świata rośnie potrzeba „chodzenia na skróty", a ewentualna strata spowodowana uproszczeniami i nieoptymalnymi decyzjami jest mniejsza niż wysiłek włożony w dogłębną analizę.

8.1 Przykłady teorii ograniczonej racjonalności

Przez lata badacze na podstawie swoich obserwacji i przeprowadzanych eksperymentów, proponowali różne modele ograniczonej racjonalności. Skupione są one na różnych aspektach niedoskonałości ludzkich możliwości poznawczych, wskutek tego niemożliwe jest ich porównanie i wybranie jednego optymalnego modelu odzwierciedlającego sposób podejmowania decyzji przez ludzi. Wybór najlepszego modelu często zależy od postawionego problemu, a nawet okoliczności, w których jest on rozwiązywany czy grupy osób, które podejmują decyzje. Z tego względu nie ma konsensusu, który model ograniczonej racjonalności jest najlepszy. W praktyce jednak część modeli zyskała

większe uznanie i cieszy się wyższą popularnością z uwagi na sukcesy ich zastosowań praktycznych. Te właśnie modele będą zaprezentowane i rozważane w rozprawie.

8.1.1 Teoria zakotwiczenia

Teoria zakotwiczenia (ang. *Anchoring Theory*) [73] postuluje, że ludzie mają tendencję do spłaszczania prawdopodobieństw przedstawianych zdarzeń. W procesie decyzyjnym rozkład prawdopodobieństwa odbierają jako bliższy rozkładowi jednostajnemu niż w rzeczywistości. Oznacza to, że wysokie prawdopodobieństwa zdarzeń są postrzegane jako mniejsze, a wartości małe odbierane są jako wyższe. Formalnie tę zależność można zapisać jako:

$$p'(x) = p(x)(1-\delta) + \frac{\delta}{|X|}, \tag{8.1}$$

gdzie X jest zbiorem wszystkich zdarzeń, $|X|$ oznacza liczność tego zbioru, $p(x)$ jest rzeczywistym prawdopodobieństwem zdarzenia $x \in X$, $p'(x)$ jest prawdopodobieństwem zaburzonym (postrzeganym przez ludzi zgodnie z teorią zakotwiczenia), a $\delta \in [0,1]$ jest liczbą rzeczywistą będącą parametrem decydującym o sile zaburzenia. Dla $\delta = 0$: $p' = p$, czyli postrzegane prawdopodobieństwa są równe tym rzeczywistym, natomiast dla $\delta = 1$: $\forall_{x \in X} p'(x) = \frac{1}{|X|}$, co oznacza rozkład jednostajny. W rozprawie przyjęto wartość najczęściej spotykaną w praktyce $\delta = 0.5$.

8.1.2 Teoria perspektywy

Teoria perspektywy (ang. *Prospect Theory*) [31] bazuje na spostrzeżeniu, że niechęć do ponoszenia strat i chęć zysku są asymetryczne. W przeprowadzonych eksperymentach ludzie wykazywali dużą awersję do podejmowania zakładów, które mogłyby im przynieść znaczne straty nawet w obliczu możliwości wielkiego zysku. Byli skłonni do brania udziału w zakładach o mniejszym ryzyku straty mimo niższej wartości oczekiwanej. Przykładowy eksperyment został zaprezentowany w tabeli 8.1. Przedstawione opcje posiadają taką samą wartość oczekiwaną, lecz wybory badanych osób były różne w zależności od kwoty, prawdopodobieństwa oraz wariantu (zysk czy strata).

W serii eksperymentów psychologicznych pokazano, że zamiast maksymalizować oczekiwaną wypłatę, ludzie podświadomie maksymalizują inną wartość nazwaną *per-*

Opcja A	Opcja B	Wybór badanych
50% szansa na zysk 1000£	zysk w wysokości 500£	B
50% szansa na stratę 1000£	strata w wysokości 500£	A
0,1% szansa na zysk 5000£	zysk w wysokości 5£	A
0,1% szansa na stratę 5000£	strata w wysokości 5£	B

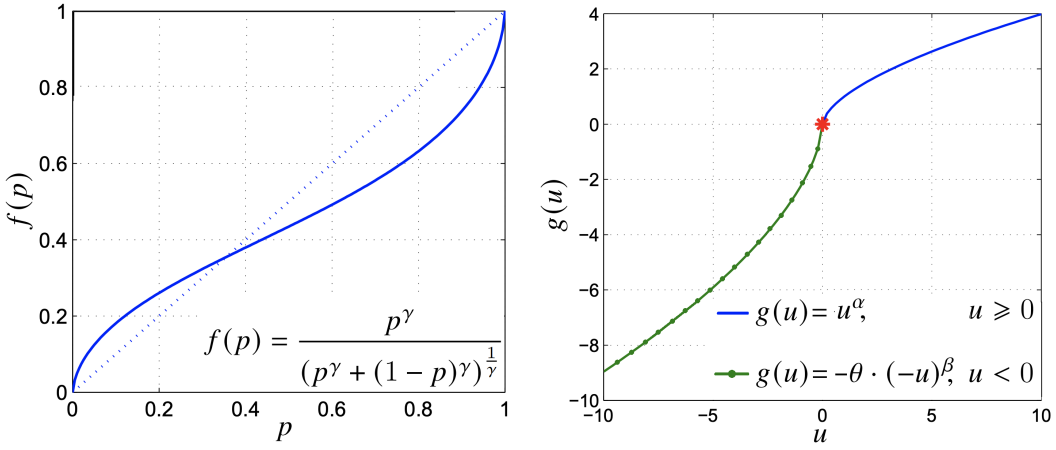
Tabela 8.1: Przykładowy eksperyment obrazujący działanie teorii perspektywy.

spektywą (P), którą można zapisać wzorem:

$$P = \sum_i f(p_i)g(u_i), \tag{8.2}$$

$$f(p_i) = \frac{p_i^\gamma}{(p_i^\gamma + (1-p_i)^\gamma)^{\frac{1}{\gamma}}}, \quad g(u_i) = \begin{cases} u_i^\alpha & \text{dla } u_i \geqslant 0 \\ -\theta \cdot (-u_i)^\beta, & \text{dla } u_i < 0 \end{cases}$$

f i g są funkcjami przekształcającymi postrzeganie rzeczywistego prawdopodobieństwa p_i otrzymania wypłaty u_i, a γ, θ, α i β są parametrami. W rozprawie przyjęto następujące wartości tych parametrów: $\gamma = 0.64, \theta = 2.25, \alpha = \beta = 0.88$ (zgodnie z rekomendacjami przedstawionymi w [74]). Wizualizację tak sparametryzowanych funkcji f oraz g prezentuje rysunek 8.1.



Rysunek 8.1: Funkcje prezentujące postrzeganie prawdopodobieństwa (z lewej) oraz wypłaty (z prawej) zgodnie z teorią perspektywy.

8.1.3 Teoria kwantowej odpowiedzi

Teoria kwantowej odpowiedzi (ang. *Quantal Response*) [46] głosi, że ludzie podejmują decyzje w sposób losowy, ale zależny od wypłaty, tzn. im większa jest wypłata związana

z podjęciem jakiejś decyzji, tym generalnie wyższe prawdopodobieństwo jej wyboru. Oznacza to, że optymalna decyzja ma największe prawdopodobieństwo wyboru, ale każda inna decyzja również może być podjęta. Zgodnie z tą teorią prawdopodobieństwo podjęcia decyzji x_i wyrażone jest wzorem:

$$p(x_i) = \frac{e^{\lambda u(x_i)}}{\sum_{x_j \in \Pi} e^{\lambda u(x_j)}}, \tag{8.3}$$

gdzie $u(x_i)$ jest oczekiwaną wypłatą w przypadku wyboru decyzji x_i, Π jest zbiorem wszystkich możliwych decyzji, a $\lambda \geqslant 0$ jest parametrem. Jeśli $\lambda = 0$, to wybór jest w pełni losowy - każda decyzja jest jednakowo prawdopodobna. Dla $\lambda \to \infty$ podejmowana jest na pewno decyzja optymalna - w pełni racjonalny wybór. W eksperymentach opisanych w niniejszej rozprawie przyjęto $\lambda = 0.8$.

8.1.4 Reguła satysfakcji

Reguła satysfakcji [68] zakłada, że wybierany jest pierwszy wariant, który spełnia pewne minimalne oczekiwania co do rezultatu, bez rozważania kolejnych opcji. Innymi słowy, podejmowana jest losowa decyzja spośród wszystkich decyzji, które nie są gorsze niż pewien ustalony ε od wypłaty związanej z optymalnym wyborem. Często regułę tą nazywa się ε optymalnością (ang. ε-*optimality*).

8.2 Ograniczona racjonalność w grach obronnych Stackelberga

Jak zostało wspomniane na początku tego rozdziału, gry obronne Stackelberga, ze względu na silne umotywowanie praktyczne oraz specyficzną grupę potencjalnych graczy wcielających się w rolę atakujących (np. kłusownicy, terroryści, złodzieje), są obszarem, w którym skorzystanie z teorii ograniczonej racjonalności może przynieść wiele korzyści. Możliwość ta została dostrzeżona już w kilku pracach, które implementują wybrane teorie ograniczonej racjonalności do gier obronnych Stackelberga.

Jednym z najbardziej znanych rozwiązań jest algorytm COBRA [61], który modyfikuje program liniowy DOBSS [59] (opisany w rozdziale 4.1.1) w celu uwzględnienia teorii zakotwiczenia wraz z ε optymalnością. Podobne podejście zostało przedstawione

w pracy [85], w której zaproponowane zostały dwa rozwiązania oparte o zmodyfikowane programy liniowe z zaaplikowanymi teoriami perspektywy i zakotwiczenia. Przeprowadzone eksperymenty potwierdziły użyteczność takiego podejścia w zastosowaniach, w których w rolę graczy wcielają się ludzie. Uwaga badaczy projektujących kolejną z metod nazwaną SHARP [32] skupiona była na specyficznych aspektach gry takich jak np. rola wcześniejszych rozgrywek i doświadczenia graczy. W pracy przedstawiono ogólny wzór modyfikujący (w sposób liniowy) wypłatę postrzeganą przez graczy, a jego parametry były ustalane eksperymentalnie. System MATCH [62] z kolei optymalizuje strategię Obrońcy, przyjmując najgorszą (dla Obrońcy) strategię Atakującego, który podejmuje decyzję zgodnie z regułą satysfakcji. Natomiast w podejściu BRQR [84] zaimplementowana została teoria kwantowej odpowiedzi, a metoda ta została następnie rozszerzona (SU-BRQR [56]) o pewną z góry założoną funkcję modyfikującą postrzegane wypłaty, której parametry zostały dobrane w drodze eksperymentów z uczestnictwem ludzi.

Wszystkie wyżej wymienione podejścia dotyczyły wdrożeń teorii ograniczonej racjonalności do gier jednokrokowych. Większość z nich oparta jest na modyfikacji sformułowanych wcześniej programów liniowych. Podobna implementacja tych teorii w wariancie wielokrokowym jest niemożliwa ze względu na pojawienie się nieliniowych zależności, które nie mogą być opisane przy pomocy programu liniowego. Być może z tego względu w literaturze brakuje metod, które uwzględniałyby teorie ograniczonej racjonalności w wielokrokowych grach obronnych Stackelberga. Odpowiedzią na tę lukę może być rozszerzenie zaprezentowanego w poprzednich rozdziałach algorytmu ewolucyjnego o możliwość uwzględnienia ograniczonej racjonalności graczy.

8.3 Rozszerzenie algorytmu ewolucyjnego

Implementacja teorii ograniczonej racjonalności w metodach dotychczas spotykanych w literaturze często nie była prosta. Wymagała zazwyczaj wprowadzenia wielu zmian w algorytmie, pewnych uproszczeń, przybliżeń czy nawet modyfikacji samej teorii ograniczonej racjonalności. Każda z tak zmodyfikowanych metod była w stanie działać jedynie z jedną wybraną teorią, zgodnie z którą nastąpiła modyfikacja algorytmu. Pre-

zentowane rozwiązania były nieelastyczne.

Powyższe słabości nie dotyczą algorytmu EASG. Jego rozszerzenie o dowolną z teorii ograniczonej racjonalności jest bardzo proste. Zmiany wymaga jedynie procedura ewaluacji rozwiązań, w której wyznaczana jest odpowiedź Atakującego na strategię Obrońcy reprezentowaną przez ocenianego osobnika. Przypomnijmy, że odpowiedź Atakującego wyznaczana jest poprzez sprawdzenie po kolei wszystkich jego możliwych strategii prostych. Dla każdej z tych strategii liczony jest rezultat gry (wypłaty graczy) w przypadku zagrania tej strategii (przez Atakującego) oraz strategii Obrońcy zakodowanej w chromosomie. Wystarczy więc w tym miejscu zmodyfikować tę strategię Obrońcy zgodnie z wybraną teorią ograniczonej racjonalności, tj. przy pomocy odpowiednich wzorów obliczyć wypłaty i prawdopodobieństwa, jakie będzie postrzegał Atakujący stosownie do implementowanej teorii. Algorytm 8.1 prezentuje zmodyfikowaną funkcję ewaluacji strategii Obrońcy. Zmiany w stosunku do wersji bez ograniczonej racjonalności można zauważyć, porównując przedstawiony fragment pseudokodu z algorytmem 5.2.

W ten sposób można rozszerzyć algorytm EASG o dowolną teorię ograniczonej racjonalności, co potwierdza uniwersalność zaproponowanego podejścia ewolucyjnego.

Algorytm 8.1: Ewaluacja strategii Obrońcy π_O z uwzględnieniem ograniczonej racjonalności atakującego.

```
1 EvaluateSolutionBR (π_O)
2     σ_bestA ← null
3     for σ_A ∈ Σ_A do
4         π*_O ← bounded_rationality(π_O)
5         if σ_bestA = null or u_A(π*_O, σ_A) > u_A(π*_O, σ_bestA) or
6         (u_A(π*_O, σ_A) = u_A(π*_O, σ_bestA) and u_O(π*_O, σ_A) > u_O(π*_O, σ_bestA)) then
7             σ_bestA ← σ_A
8     return u_O(π_O, σ_bestA)
```

Porównywanie wyników (wypłat Obrońcy) policzonych z uwzględnieniem ograniczonej racjonalności i bez niej jest bezcelowe, ponieważ są to de facto dwa różne problemy. Wiemy, że w przypadku ograniczonej racjonalności Atakującego jego decyzje nie muszą być optymalne, ale jednocześnie zakładamy, że Obrońca o tym wie i odpowiednio modyfikuje swoją strategię. Ta "nieoptymalność" Atakującego może spowodować zarówno

wyższą, jak i niższą oczekiwaną wypłatę Obrońcy. Niemożliwym jest więc uniwersalne stwierdzenie, jaka powinna być relacja wypłat z ograniczoną racjonalnością i bez niej oraz co otrzymana różnica będzie oznaczała w kontekście poprawności czy jakości algorytmu. Dlatego też weryfikacja zaproponowanego powyżej rozszerzenia EASG o ograniczoną racjonalność Atakującego zostanie zaprezentowana na końcu tego rozdziału w kontekście porównania z kolejnym podejściem opartym o sieci neuronowe. Czas działania algorytmu nie ulega znaczącej zmianie. Zależy on od implementacji konkretnego modelu ograniczonej racjonalności, lecz w większości przypadków te dodatkowe obliczenia są pomijalne w kontekście czasu działania całego algorytmu ewolucyjnego.

8.4 Sieć neuronowa w EASG

8.4.1 Motywacja

Jak zostało zauważone we wstępie tego rozdziału gry obronne Stackelberga są używane do modelowania scenariuszy ochrony zasobów przed terrorystami, złodziejami, czy kłusownikami. W praktyce przeciwnicy nie są dobrze znani obrońcom - trudno ocenić ich preferencje, poziom wiedzy czy umiejętności. W dotychczasowych rozważaniach zakładaliśmy pełną wiedzę obu graczy o problemie - w szczególności Obrońca zna dokładną wartość wypłaty Atakującego dla każdego z celów i na tej podstawie może przewidzieć, który z nich zostanie zaatakowany. W rzeczywistości możemy tylko szacować te wartości na podstawie wiedzy eksperckiej czy obserwacji wcześniejszych ataków lub zachowań atakujących. Również wdrożenie poruszanej w tym rozdziale idei ograniczonej racjonalności może w praktyce stanowić wyzwanie.

Dotychczas prezentowane w literaturze rozwiązania (jak również zademonstrowane wcześniej w tej rozprawie podejście ewolucyjne) z góry przyjmują jeden konkretny model ograniczonej racjonalności i zgodnie z nim wyliczają najlepszą strategię Obrońcy. Jednak nie istnieje jeden, wyróżniony, najlepszy model ograniczonej racjonalności, którego wdrożenie można zarekomendować w każdej sytuacji. W takim przypadku również trzeba kierować się intuicją, wiedzą lub doświadczeniem pozyskanymi z innego źródła. Potrzeba zatem rozwiązania, które potrafiłoby znajdować dobre strategie Obrońcy, biorąc pod uwagę wyżej wymienione ograniczenia pojawiające się w rzeczywistych sce-

nariuszach, tj. brak pełnej wiedzy o Atakującym, jego preferencjach i najlepiej dopasowanym modelu ograniczonej racjonalności. W dalszej części tego rozdziału zostanie zaproponowana metoda odpowiadająca tej potrzebie.

8.4.2 Przykładowy scenariusz - cyberbezpieczeństwo

Gra, która będzie rozważana w eksperymentach, inspirowana jest problemem z obszaru cyberbezpieczeństwa nazywanym *głęboką inspekcją pakietów* (ang. *Deep Packet Inspection*) [22]. Jest to technika sieciowa, przy pomocy której analizowane są pakiety przesyłane przez sieć, w celu wykrycia w nich anomalii świadczących o możliwym ataku. Dokładna analiza pakietów jest kosztowna i powoduje opóźnienia w przesyle danych, dlatego nie może być wykonywana w sposób ciągły w całej sieci. W praktyce wybierany jest jakiś podzbiór hostów, dla których przeprowadzana jest analiza pakietów. Sytuację tę można naturalnie zamodelować jako grę obronną Stackelberga, w której system bezpieczeństwa, decydujący o inspekcji pakietów, pełni rolę Obrońcy, a potencjalni przestępcy internetowi odgrywają rolę Atakujących. Strategią Obrońcy jest wybór hostów, których pakiety będą poddane analizie. Strategia Atakującego polega na wyborze jednego hosta do przeprowadzenia ataku. Dla każdego hosta przypisane są dwie wypłaty Obrońcy: związana z wykryciem intruza oraz będąca skutkiem udanego ataku. Obrońca nie zna potencjalnego intruza, więc nie posiada informacji o jego preferencjach i wypłatach związanych z atakiem poszczególnych celów.

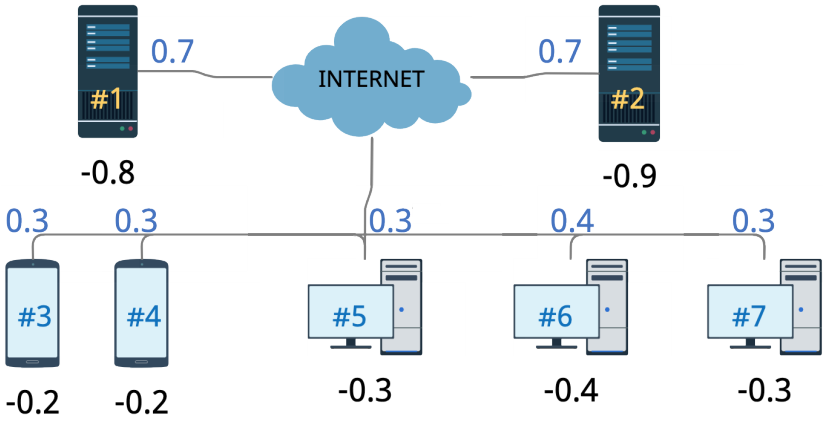
Bardziej sformalizowany opis przyjętego w tym rozdziale modelu gry jest następujący. Zgodnie z oznaczeniami z rozdziału 3.3 dany jest zbiór celów (hostów) T o liczności n. Do każdego z nich przypisane są 4 wypłaty U_t^j, $j \in \{O+, O-, A+, A-\}$. Gra rozgrywana jest w m krokach czasowych. W każdym z nich Obrońca wybiera podzbiór celów o liczności k (dla których przeprowadza głęboką inspekcję pakietów). Zatem strategię prostą Obrońcy można zapisać jako $\sigma^O = \{a_1, a_2, \ldots, a_m\}$, gdzie $a_i \subseteq T$ jest podzbiorem celów "chronionych" w i-tej jednostce czasu. Strategią Atakującego jest wybór jednego celu - oznaczmy go przez t_x.

Wypłaty graczy liczone są następująco:
- Jeśli w jakimkolwiek kroku czasowym jednostka Obrońcy jest przypisana do celu t_x , to Atakujący jest *schwytany*, a gracze otrzymują odpowiednio wypłaty $U_{t_x}^{O^+}$ oraz $U_{t_x}^{A^-}$.

- Jeśli nie istnieje żaden krok czasowy, w którym jakakolwiek jednostka Obrońcy jest przypisana do celu t_x, to atak jest udany, a gracze otrzymują odpowiednio wypłaty $U_{t_x}^{O^-}$ oraz $U_{t_x}^{A^+}$.

Zatem oczekiwana wypłata Obrońcy (u^O) i Atakującego (u^A) wynoszą odpowiednio
$u^O = P_{t_x} U_{t_x}^{O-} + (1 - P_{t_x}) U^{O+}$ oraz $u^A = P_{t_x} U_{t_x}^{A+} + (1 - P_{t_x}) U_{t_x}^{A-}$,
gdzie $P_{t_x} = \prod_{s=1,\ldots,m}(1 - c_s(x))$ jest prawdopodobieństwem udanego ataku na cel t_x. Przypomnijmy, że zgodnie z notacją wprowadzoną w rozdziale 3.3 przez $c_s(x)$ oznaczone jest pokrycie celu, czyli prawdopodobieństwo, że przynajmniej jedna jednostka Obrońcy przypisana jest do celu x w kroku czasowym s.

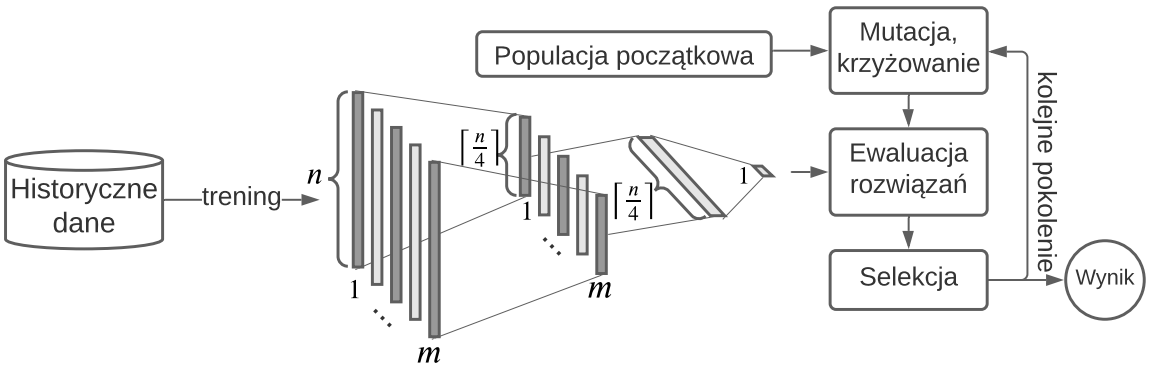
Przykładowa gra z $n = 7$ hostami przedstawiona jest na rysunku 8.2. Wypłaty Obrońcy w przypadku udanego ataku U^{O-} na dany cel znajdują się pod każdym z nich. Dla uproszczenia przyjęto zerową wypłatę w sytuacji udaremnienia ataku ($U^{O+} = 0$) oraz jeden krok czasowy ($m = 1$). Sieć zawiera 2 serwery z danymi (które są kluczowymi elementami infrastruktury), 3 komputery i 2 urządzenia mobilne. Zakładamy, że Obrońca może analizować jednocześnie pakiety docierające do maksymalnie $k = 3$ hostów. Rysunek prezentuje również strategię Obrońcy w postaci prawdopodobieństwa pokrycia każdego z celów - $c_s(t)$. Przykładowa strategia mieszana, która odpowiada przedstawionemu pokryciu to: $\pi^O = \{(0.4, \{\#1, \#2, \#6\}), (0.3, \{\#1, \#2, \#7\}), (0.3, \{\#3, \#4, \#5\})\}$. Jeśli przykładowo Atakujący wybierze do ataku cel #5, to oczekiwana wypłata Obrońcy wyniesie $-0.3(1 - 0.3) = -0.21$.



Rysunek 8.2: Przykładowy scenariusz w problemie głębokiej analizy pakietów.

8.4.3 Architektura rozwiązania

Proponowane rozwiązanie oparte jest o wykorzystanie sztucznej sieci neuronowej do oceny jakości strategii Obrońcy. Sieć neuronowa zastępuje najbardziej kosztowną obliczeniowo część metody EASG - ewaluację osobników, która każdorazowo wymaga iterowania po wszystkich możliwych strategiach Atakującego. Sieć neuronowa aproksymuje oczekiwaną wypłatę Obrońcy na podstawie jego strategii. Zwrócona przez sieć wartość jest przypisywana do ocenianego osobnika jako wartość jego funkcji przystosowania. Schemat architektury proponowanego rozszerzenia metody EASG o sieć neuronową zaprezentowany jest na rysunku 8.3.



Rysunek 8.3: EASG rozszerzone o sieć neuronową do oceny rozwiązań.

Wybraną architekturą sieci jest perceptron wielowarstwowy z dwiema warstwami ukrytymi. Strategia Obrońcy podawana na wejściu sieci neuronowej kodowana jest w następujący sposób. Każdy z neuronów wejściowych przyjmuje prawdopodobieństwo pokrycia celu $c_s(t)$. Wartość ta wynika wprost z ocenianej strategii Obrońcy (patrz rozdział 3.3) i może być w prosty sposób wyliczona analitycznie. Implikuje to rozmiar wejściowy sieci: mn (liczba kroków czasowych $\times$ liczba celów). Neurony z pierwszej (wejściowej) warstwy sieci są grupowane według kroków czasowych - każdemu krokowi czasowemu odpowiada $\lceil \frac{n}{4} \rceil$ neuronów w drugiej warstwie, tj. każde n neuronów z warstwy wejściowej połączonych jest (każdy z każdym) z odpowiednią grupą $\lceil \frac{n}{4} \rceil$ w warstwie drugiej. Zatem liczność warstwy drugiej wynosi $\lceil \frac{n}{4} \rceil m$. Następnie wszystkie neurony z tej warstwy połączone są (każdy z każdym) z neuronami warstwy trzeciej o liczności $\lceil \frac{n}{4} \rceil$. Sygnał z nich trafia ostatecznie do pojedynczego neuronu w warstwie

wyjściowej, który zwraca jedną liczbę rzeczywistą, będącą estymowaną wartością wypłaty Obrońcy dla strategii podanej na wejściu sieci. Sieć neuronowa uczona jest przy pomocy historycznych danych, czyli wyników poprzednich rozgrywek.

Zaprezentowany perceptron wielowarstwowy zastępuje procedurę ewaluacji rozwiązań w algorytmie EASG opisanym w rozdziale 5. Wszystkie pozostałe elementy algorytmu pozostają bez zmian. W dalszej części algorytm EASG rozszerzony o sieć neuronową będzie nazywany NESG od angielskiego określenia *NeuroEvolutionary for Security Games*.

8.4.4 Eksperymenty

Metoda NESG została przetestowana na 90 losowych grach odpowiadających scenariuszowi z obszaru cyberbezpieczeństwa opisanym w rozdziale 8.4.2. Stworzono 5 różnych gier dla każdej z $m \in \{1, 2, 4\}$ liczby kroków czasowych i liczby celów $n = 2^i$, $i \in \{2, \ldots, 7\}$. Wypłaty U_t^{O-} i U_t^{A-} losowane były (niezależnie dla każdego celu) z przedziału $(-1, 0)$, natomiast U_t^{O+} i U_t^{A+} z przedziału $(0, 1)$. Liczba dostępnych jednostek Obrońcy również była wartością losową (dla każdej instancji gry ustalana niezależnie) z przedziału $[\lfloor \frac{n}{4m} \rfloor, \lceil \frac{3n}{4m} \rceil]$, co oznacza, że co najmniej $\frac{1}{4}$ i co najwyżej $\frac{3}{4}$ z celów może być efektywnie chronionych.

W eksperymentach przyjęto identyczne wartości parametrów jak dla algorytmu EASG (przedstawione w tabeli 6.1 w rozdziale 6).

Sieć neuronowa była trenowana metodą propagacji wstecznej błędów z optymalizatorem Adam [38]. W warstwie wyjściowej użyto tangensa hiperbolicznego jako funkcji aktywacji neuronu, w pozostałych warstwach było to ReLU [51]. Aby zasymulować dane dotyczące historycznych rozgrywek służących do nauki sieci, wygenerowano 5000 przykładów. Każdy przykład został stworzony według następującej procedury. Strategię mieszaną Obrońcy (dane wejściowe sieci) wybrano losowo spośród wszystkich możliwych strategii złożonych z co najwyżej 5 strategii prostych: najpierw wylosowano liczbę tych strategii l ze zbioru $\{1, \ldots, 5\}$, następnie dla każdej z l strategii prostych losowano podzbiór k celów, a prawdopodobieństwo danej strategii prostej było wyznaczane jako losowa liczba z przedziału $[0, 1]$. Finalnie prawdopodobieństwa wszystkich wygenerowanych w ten sposób strategii prostych, wchodzących w skład strategii mieszanej,

zostały znormalizowane, aby ich suma wynosiła 1. Dla tak wygenerowanej strategii mieszanej Obrońcy wyznaczana była optymalna odpowiedź Atakującego i wyliczana oczekiwana wypłata Obrońcy zgodnie z algorytmem 8.1. Wynik ten był oczekiwaną wartością wyjścia sieci używaną w treningu.

W eksperymentach testowane były 3 różne modele ograniczonej racjonalności: teoria zakotwiczenia (TZ), teoria kwantowej odpowiedzi (TKO) oraz teoria perspektywy (TP). Informacja, który z tych modeli został zastosowany do wygenerowania danych nie była wprost podana na żadnym etapie działania algorytmu - jedynie przy pomocy zadanej teorii wyliczana była odpowiedź Atakującego w danych treningowych, co miało wpływ na wypłaty Obrońcy podawane w procesie nauki sieci jako historyczne dane. Eksperyment sprawdzał, czy sieć neuronowa tylko na podstawie takich informacji (bez implementacji wprost żadnego z modeli ograniczonej racjonalności oraz podania wypłat Atakującego) będzie zdolna dobrze estymować wypłatę Obrońcy i jak ewentualne niedokładności tej estymacji wpłyną na jakość znajdowanych strategii w całym procesie ewolucji.

8.4.5 Wyniki

Sieć neuronowa

Tabela 8.2 przedstawia średni błąd (liczony jako wartość bezwzględna z różnicy wartości otrzymanej i oczekiwanej) zwracanej przez sieć neuronową wypłaty Obrońcy na zbiorze testowym. Sieć testowana była metodą 10-krotnej walidacji krzyżowej. Na podstawie zaprezentowanych danych można wysnuć dwa wnioski. Po pierwsze błąd estymacji sieci rośnie wraz ze wzrostem liczby celów i kroków czasowych, co jest spodziewanym rezultatem ze względu na zwiększenie poziomu skomplikowania problemu i większą liczbę danych wejściowych sieci. Po drugie można zaobserwować pewne różnice dokładności estymacji pomiędzy poszczególnymi modelami ograniczonej racjonalności. Sieć dokładniej przybliża wartości dla teorii zakotwiczenia oraz kwantowej odpowiedzi niż dla teorii perspektywy. Prawdopodobną przyczyną tego zjawiska jest fakt, że teoria perspektywy w sposób nieliniowy wpływa zarówno na zmianę postrzeganych wypłat jak i prawdopodobieństwa, podczas gdy pozostałe modele zmieniają tylko jeden z tych parametrów:

teoria zakotwiczenia prawdopodobieństwo, a teoria kwantowej odpowiedzi wypłaty.

	t. zakotwiczenia			t. kwantowej odpowiedzi			t. perspektywy		
l. celów	1 krok	2 kroki	3 kroki	1 krok	2 kroki	3 kroki	1 krok	2 kroki	3 kroki
4	0.006	0.006	0.006	0.004	0.005	0.005	0.010	0.010	0.010
8	0.011	0.012	0.014	0.008	0.008	0.008	0.018	0.019	0.020
16	0.024	0.026	0.028	0.019	0.021	0.022	0.046	0.049	0.051
32	0.043	0.045	0.048	0.031	0.033	0.035	0.075	0.080	0.084
64	0.080	0.081	0.086	0.064	0.065	0.069	0.132	0.142	0.145
128	0.119	0.125	0.131	0.104	0.110	0.121	0.232	0.241	0.251

Tabela 8.2: Średni błąd zwracanej przez sieć wypłaty Obrońcy na zbiorze testowym.

Wypłaty

Na podstawie samych wartości błędów odpowiedzi zwracanych przez sieć trudno stwierdzić, czy otrzymywane rezultaty są satysfakcjonujące i jak wpłyną na jakość rozwiązań pierwotnego problemu, czyli poszukiwania optymalnej strategii Obrońcy z równowagi Stackelberga. W celu przeprowadzenia takiej oceny metodę NESG porównano z pięcioma następującymi metodami: **C2016** - metoda dokładna opisana w rozdziale 4.1.3, nieuwzględniająca ograniczonej racjonalności Atakującego, **EASG** - algorytm ewolucyjny zaproponowany w rozdziale 5 bez dodanego modelu ograniczonej racjonalności oraz **EASG_BR**, gdzie BR $\in$ {TZ, TKO, TP} - metoda EASG z dodaną implementacją modelu ograniczonej racjonalności BR zgodnie z algorytmem 8.1.

W przypadku metod C2016 i EASG najpierw policzona została za ich pomocą strategia Obrońcy (bez zakładania ograniczonej racjonalności przeciwnika), a następnie wypłata Obrońcy (grającego według zwróconej przez metodę strategii) wyznaczona została z uwzględnieniem odpowiedniego modelu ograniczonej racjonalności. Należy zauważyć, że przytoczone modele ograniczonej racjonalności nie mogą być wprost włączone do algorytmu dokładnego opartego o programowanie liniowe (C2016) ze względu na ich nieliniową charakterystykę [36]. Zatem nie jest możliwe w prosty sposób (bez istotnej modyfikacji metody lub modelu ograniczonej racjonalności) policzenie optymalnej strategii Obrońcy uwzględniającej wybrany model ograniczonej racjonalności.

Tabela 8.3 prezentuje porównanie wyżej wymienionych metod pod względem średniej wypłaty Obrońcy. We wszystkich przypadkach proponowany algorytm NESG osiągnął lepsze wyniki niż obie metody, które nie zakładały żadnej z teorii ograniczonej

racjonalności (C2016 i EASG). Przewaga ta zwiększa się wraz z liczbą kroków czasowych. Na tej podstawie można wnioskować, że podczas rozgrywki przeciwko graczowi, którego decyzje nie są w pełni racjonalne, lepiej jest używać przybliżonego algorytmu NESG niż grać według optymalnej strategii (zwróconej przez C2016), która ograniczonej racjonalności nie uwzględnia.

1 krok	t. zakotwiczenia				t. kwantowej odpowiedzi				t. perspektywy			
l. celów	C2016	EASG	EASG_AT	NESG	C2016	EASG	EASG_QR	NESG	C2016	EASG	EASG_PT	NESG
4	-0.470	-0.472	-0.468	-0.469	-0.406	-0.408	-0.404	-0.405	-0.419	-0.420	-0.417	-0.418
8	-0.456	-0.457	-0.440	-0.440	-0.418	-0.422	-0.386	-0.388	-0.422	-0.423	-0.407	-0.407
16	-0.387	-0.391	-0.371	-0.371	-0.377	-0.378	-0.336	-0.338	-0.329	-0.335	-0.315	-0.318
32	-0.411	-0.412	-0.393	-0.397	-0.428	-0.429	-0.390	-0.394	-0.397	-0.404	-0.367	-0.370
64	-0.579	-0.586	-0.567	-0.568	-0.582	-0.584	-0.536	-0.537	-0.560	-0.564	-0.483	-0.486
128	-0.397	-0.405	-0.369	-0.372	-0.578	-0.578	-0.526	-0.529	-0.462	-0.463	-0.345	-0.347
2 kroki	t. zakotwiczenia				t. kwantowej odpowiedzi				t. perspektywy			
l. celów	C2016	EASG	EASG_AT	NESG	C2016	EASG	EASG_QR	NESG	C2016	EASG	EASG_PT	NESG
4	-0.566	-0.566	-0.563	-0.564	-0.540	-0.541	-0.534	-0.535	-0.548	-0.549	-0.547	-0.547
8	-0.568	-0.572	-0.553	-0.555	-0.526	-0.528	-0.510	-0.512	-0.556	-0.556	-0.517	-0.518
16	-0.327	-0.331	-0.314	-0.317	-0.326	-0.331	-0.301	-0.302	-0.326	-0.331	-0.291	-0.294
32	-0.499	-0.500	-0.475	-0.479	-0.487	-0.487	-0.435	-0.435	-0.501	-0.502	-0.454	-0.457
64	-0.457	-0.463	-0.427	-0.427	-0.421	-0.424	-0.403	-0.408	-0.466	-0.471	-0.407	-0.410
128	-0.607	-0.614	-0.563	-0.567	-0.601	-0.604	-0.540	-0.544	-0.593	-0.595	-0.566	-0.571
4 kroki	t. zakotwiczenia				t. kwantowej odpowiedzi				t. perspektywy			
l. celów	C2016	EASG	EASG_AT	NESG	C2016	EASG	EASG_QR	NESG	C2016	EASG	EASG_PT	NESG
4	-0.479	-0.481	-0.478	-0.479	-0.487	-0.489	-0.485	-0.486	-0.511	-0.512	-0.508	-0.510
8	-0.497	-0.500	-0.466	-0.467	-0.509	-0.513	-0.455	-0.456	-0.517	-0.519	-0.496	-0.499
16	-0.545	-0.547	-0.525	-0.525	-0.531	-0.534	-0.502	-0.503	-0.570	-0.574	-0.535	-0.538
32	-0.478	-0.484	-0.460	-0.464	-0.500	-0.505	-0.468	-0.470	-0.525	-0.531	-0.492	-0.496
64	-0.563	-0.568	-0.547	-0.551	-0.587	-0.593	-0.553	-0.555	-0.600	-0.600	-0.561	-0.563
128	-0.531	-0.536	-0.493	-0.497	-0.545	-0.549	-0.503	-0.505	-0.553	-0.555	-0.512	-0.512

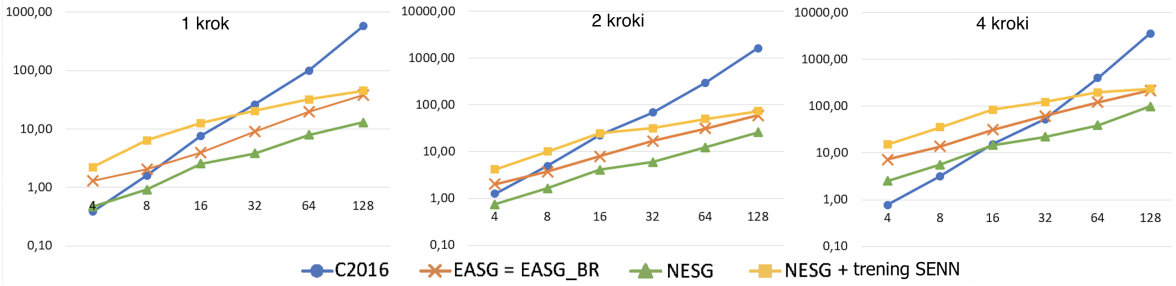
Tabela 8.3: Porównanie średnich wypłat Obrońcy uzyskanych przez poszczególne metody dla gier z 1, 2 oraz 4 krokami.

Dla danego modelu ograniczonej racjonalności BR $\in$ {TZ, TKO, TP} jedyna różnica między metodami EASG_BR oraz NESG polega na innej procedurze ewaluacji osobników. EASG_BR liczy wypłatę dokładnie na podstawie przyjętego modelu ograniczonej racjonalności, podczas gdy NESG korzysta z wcześniej nauczonej sieci neuronowej. Ze względu na fakt, że metoda EASG_BR "wie" jaki model ograniczonej racjonalności jest rozważany i wprost korzysta z jego implementacji, wyniki osiągane przez tę metodę są lepsze. Jak jednak wcześniej zauważono, nie jest to typowa sytuacja w praktyce. NESG zakłada bardziej realistyczny scenariusz, w którym Obrońca nie posiada wiedzy o zachowaniach, preferencjach czy racjonalności Atakującego. Co więcej, jego ograniczona racjonalność nie musi wpisywać się w żaden ze znanych z literatury modeli. W takiej sytuacji użycie EASG_BR nie będzie tak skuteczne jak metody NESG, która nie

przyjmuje żadnego z góry narzuconego modelu ograniczonej racjonalności.

Czas działania

Rysunek 8.4 przedstawia porównanie czasów działania poszczególnych metod. Algorytm NESG został zaprezentowany w dwóch wariantach: z uwzględnionym czasem nauki sieci oraz bez niego. Metody EASG oraz EASG_BR są zademonstrowane razem, ponieważ dodatkowy czas na wyliczenie odpowiedzi Atakującego zgodnej z wybraną teorią ograniczonej racjonalności jest pomijalny (zazwyczaj jest to kilka operacji arytmetycznych, które zmieniają wartość postrzeganej wypłaty lub prawdopodobieństwa wyboru strategii prostej).



Rysunek 8.4: Porównanie czasów działania poszczególnych metod.

Na rysunku widać, że skalowalność czasowa metod ewolucyjnych jest lepsza niż metody C2016 opartej o programowanie liniowe. EASG, EASG_BR oraz NESG skalują się w przybliżeniu w sposób liniowy względem liczby celów. Spośród tych metod najkrótszy czas obliczeń osiąga NESG. Wynika to ze sposobu ewaluacji osobników. Zamiast iteracji po wszystkich możliwych strategiach Atakującego i dla każdej z nich policzenia wypłaty (jak jest to czynione w EASG), NESG korzysta z wcześniej nauczonej sieci neuronowej. Procedura ewaluacji jest wykonywana wielokrotnie w trakcie działania algorytmu (liczba pokoleń × liczba osobników) stąd przewaga NESG w aspekcie czasu obliczeń.

Przedstawione powyżej wyniki osiąganych wypłat oraz czasu działania pozwalają stwierdzić, że zaproponowane rozszerzenie metody ewolucyjnej o sztuczną sieć neuronową estymującą wypłatę Obrońcy, może być skutecznym podejściem szczególnie w praktycznych sytuacjach, w których brakuje pełnej wiedzy o Atakującym, jego umiejętnościach i priorytetach.

Rozdział 9

Algorytm koewolucyjny

9.1 Motywacja

Zaprezentowany w rozdziale 5 algorytm EASG zakłada, że ewaluacja każdego potencjalnego rozwiązania (strategii Obrońcy) wymaga znalezienia optymalnej odpowiedzi Atakującego poprzez sprawdzenie i policzenie wypłaty dla wszystkich możliwych strategii prostych Atakującego. Jak wykazały eksperymenty, często takie podejście jest skuteczne, ale nie zawsze. W przypadku gier o innej specyfice, strategii Atakującego może być na tyle dużo, że rozważanie każdej z nich w celu ewaluacji wszystkich osobników w każdym pokoleniu byłoby zbyt czasochłonne. W szczególności możemy wyobrazić sobie grę z ciągłą przestrzenią strategii Atakującego, co w przypadku algorytmu EASG będzie oznaczać konieczność przejrzenia nieskończenie wielu strategii.

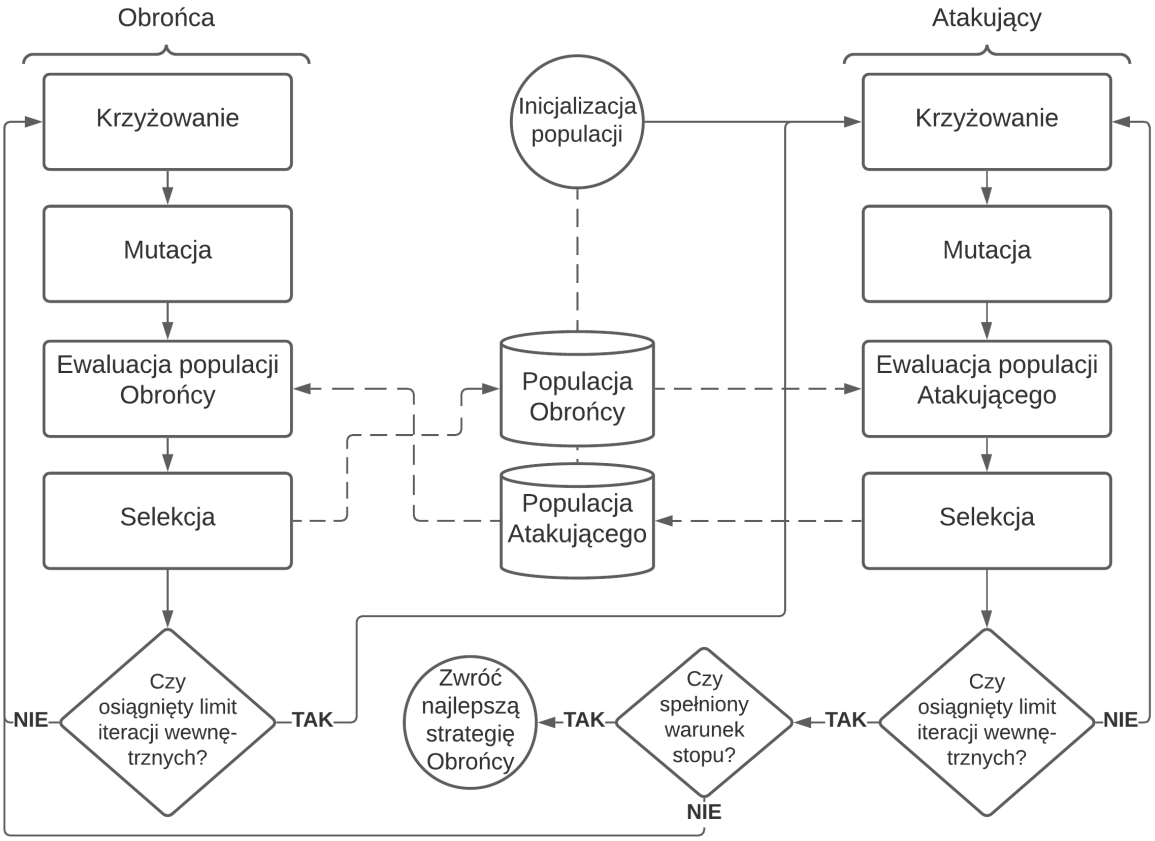
W przeprowadzonych eksperymentach zauważono, że w większości przypadków istnieje pewien niewielki podzbiór strategii Atakującego, który jest wart rozważania. Pozostałe strategie albo w trywialny sposób można określić jako słabe (np. atak na dobrze chroniony cel z niską wypłatą lub sekwencja akcji nieprowadząca do celu), albo są na tyle podobne do innych strategii, że wystarczy rozważyć jedną z nich (np. dojście do tego samego celu, ale w innym kroku czasowym). Oczywiście mogą zdarzyć się pewne szczególne przypadki, w których te strategie Atakującego będą miały kluczowe znaczenie (mogą okazać się optymalną odpowiedzią na daną strategię Obrońcy), więc nie można wykluczyć ich od razu na początku działania algorytmu. Jednak wartościowe byłoby znalezienie pewnej metody potrafiącej adaptacyjnie wybierać reprezentatywny podzbiór strategii Atakującego "wartych rozważenia" (policzenia wypłat).

Odpowiedzią na tę potrzebę jest zaproponowany w tym rozdziale algorytm koewolucyjny. Jego głównym rozszerzeniem, w stosunku do podejścia EASG, jest zbudowanie dodatkowej populacji zawierającej strategie Atakującego. Takie rozszerzenie wydaje się być naturalnym krokiem w sytuacji, kiedy model gier Stackelberga zakłada istnienie dwóch rywalizujących ze sobą graczy, a algorytmy koewolucyjne działają na zasadzie konkurujących ze sobą populacji rozwiązań (patrz rozdział 2.3.9). Populacja strategii Atakującego podlega również ewolucji (podobnie jak populacja strategii Obrońcy) i służy do ewaluacji osobników z drugiej populacji. Zamiast liczyć wypłatę Obrońcy przeciwko wszystkim możliwym strategiom Atakującego, jest ona liczona jedynie przeciwko strategiom reprezentowanym przez osobniki z drugiej populacji (Atakującego). Takie podejście zakłada, że populacje będą ze sobą w pewien sposób konkurowały. Dla znalezionych strategii Obrońcy, populacja Atakującego w toku ewolucji będzie "starała się" znaleźć jak najlepszą odpowiedź. Z drugiej strony populacja Obrońcy będzie poszukiwała strategii, które będą najbardziej skuteczne przeciwko zadanym odpowiedziom z populacji Atakującego.

9.2 Architektura rozwiązania

W proponowanym algorytmie koewolucyjnym (nazwanym CoEvoSG) tworzone są dwie populacje. Pierwsza z nich, jak w podstawowej metodzie EASG, zawiera strategie mieszane Obrońcy. Druga populacja złożona jest ze strategii prostych Atakującego. Obie inicjalizowane są losowymi strategiami, a następnie modyfikowane naprzemiennie. Najpierw populacja Atakującego rozwijana jest przy pomocy typowych dla algorytmu ewolucyjnego operatorów (krzyżowania, mutacji i selekcji) przez g_c pokoleń, po czym następuje rozwój populacji Obrońcy przez tyle samo pokoleń. Procedura ta jest powtarzana, dopóki nie zostanie spełniony warunek stopu. Schemat działania algorytmu koewolucyjnego został zaprezentowany na rysunku 9.1.

Wszystkie operatory ewolucyjne (krzyżowanie, mutacja i selekcja), działające na populacji Obrońcy, oraz warunek stopu, pozostały bez zmian względem algorytmu EASG. Ich szczegółowy opis można znaleźć w rozdziale 5. Charakterystyka pozostałych, nowych elementów zaprezentowana została poniżej.



Rysunek 9.1: Schemat działania algorytmu koewolucyjnego.

9.2.1 Krzyżowanie populacji Atakującego

W algorytmie EASG krzyżowanie polegało na łączeniu zbiorów strategii prostych wchodzących w skład zakodowanych strategii mieszanych. W przypadku strategii Atakującego to podejście musi zostać zmodyfikowane, ponieważ krzyżowane są strategie proste. Dlatego też zastosowano krzyżowanie wymieniające jednopunktowe (patrz rozdział 2.3.3). Każdy osobnik podlega krzyżowaniu z prawdopodobieństwem p_k. Osobniki wybrane do krzyżowania łączone są w pary losowo. W wyniku krzyżowania strategii $\pi_A^1 = (a_1^1, a_2^1, \ldots, a_m^1)$ i $\pi_A^2 = (a_1^2, a_2^2, \ldots, a_m^2)$ powstają dwie strategie:
$\pi_A'^1 = (a_1^1, a_2^1, \ldots a_i^1, a_{i+1}^2, \ldots, a_m^2)$ oraz $\pi_A'^2 = (a_1^2, a_2^2, \ldots a_i^2, a_{i+1}^1, \ldots, a_m^1)$, gdzie $a_i^1 = a_i^2$ jest pierwszą wspólną akcją (w tym samym kroku czasowym) krzyżowanych strategii (np. w przypadku gier WHG, SEG czy FIG będzie to pierwszy wspólny wierzchołek odwiedzany w tym samym kroku czasowym). Jeśli taka akcja nie istnieje, to strategie pozostają bez zmian - krzyżowanie nie wywołuje żadnych efektów.

9.2.2 Mutacja populacji Atakującego

Mutacja populacji Atakującego działa analogicznie jak mutacja w algorytmie EASG (rozdział 5.5). Każdy osobnik podlega mutacji z określonym prawdopodobieństwem (p_m). W przypadku Atakującego w osobnikach zakodowane są strategie proste, które w nietrywialnym[1] przypadku składają się z ciągu akcji w kolejnych krokach czasowych. Procedura mutacji wybiera losowy krok czasowy i, począwszy od tego kroku, modyfikuje kolejne akcje w strategii Atakującego. Każda kolejna akcja wybierana jest losowo spośród wszystkich dostępnych akcji w danym stanie.

9.2.3 Selekcja populacji Atakującego

Do selekcji strategii Atakującego, które zostaną przeniesione do nowego pokolenia użyto tego samego podejścia co w algorytmie EASG, czyli turnieju binarnego z presją selekcji poprzedzonego procedurą elitaryzmu. Szczegóły zostały opisane w rozdziale 5.9.

9.2.4 Ewaluacja

Najbardziej istotną zmianą względem algorytmu EASG jest modyfikacja procedury ewaluacji rozwiązań. Podczas ewaluacji strategii Obrońcy, zamiast obliczać wypłaty dla każdej możliwej strategii prostej Atakującego, optymalna odpowiedź poszukiwana jest wśród strategii zakodowanych w populacji Atakującego. Może się zdarzyć, że istnieje strategia, będąca optymalną odpowiedzią, która nie należy do populacji Atakującego. W takim przypadku oczekiwanym wynikiem działania algorytmu jest stworzenie jej w procesie ewolucji w kolejnych pokoleniach.

Sposób ewaluacji osobników z populacji Atakującego nie jest tak oczywisty jak w przypadku Obrońcy. Zazwyczaj nie istnieje jedna najlepsza uniwersalna odpowiedź Atakującego na wszystkie strategie Obrońcy. Raczej w zależności od przyjętej strategii Obrońcy, optymalna strategia Atakującego zmienia się. Oczekiwaną sytuacją jest posiadanie w populacji wszystkich strategii Atakującego, które są optymalnymi odpowiedziami dla jakiejś strategii Obrońcy. Zatem przyjęcie średniej wypłaty Atakującego

[1] W przypadku niektórych gier (np. SGP) strategia Atakującego polega jedynie na wyborze celu do ataku. W takiej sytuacji mutacja polega na zmianie tego celu na losowy inny, a krzyżowania nie stosuje się.

względem całej populacji Obrońcy (lub jej części) może być pomysłem nietrafionym, bowiem dana strategia Atakującego może być kluczowa (optymalna) tylko w konkretnym przypadku (dla jednej strategii Obrońcy), a przyjęcie średniej obniży wartość przystosowania takiej strategii.

Zatem lepszym podejściem jest przyjęcie miary maksimum. W populacji Obrońcy (w celu zachowania jej różnorodności) istnieją strategie, które są słabe. Dla nich większość strategii Atakującego będzie dawała wysokie wypłaty, co również podważa użycie jako funkcji przystosowania maksimum wypłaty Atakującego względem wszystkich osobników populacji Obrońcy. Dlatego też wybrany został wariant pośredni - przystosowaniem osobników z populacji Atakującego jest maksimum wypłaty Atakującego względem N_{top} najlepszych (najlepiej przystosowanych) strategii z populacji Obrońcy. Wartość parametru N_{top} jest ustalana eksperymentalnie.

9.3 Eksperymenty

9.3.1 Parametry

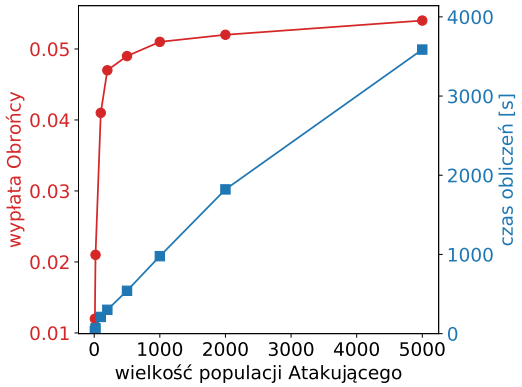
W przeprowadzonych eksperymentach przyjęto te same wartości parametrów, które były używane w testach metody EASG (patrz tabela 7.1). Dla prawdopodobieństwa mutacji i krzyżowania oraz presji selekcji w populacji Atakującego ustalono te same wartości jak w populacji Obrońcy. Algorytm koewolucyjny wprowadza kilka dodatkowych parametrów, które nie były wcześniej testowane. W celu ich zbadania przeprowadzono szereg eksperymentów, wykorzystując zbiór 50 losowo wygenerowanych gier WHG, które nie były następnie używane w procesie ewaluacji metody CoEvoSG.

Pierwszym z badanych parametrów była wielkość populacji Atakującego (N_A). Wyniki zostały zaprezentowane na rysunku 9.2a. Im większa liczba osobników w populacji Atakującego tym dokładniejsza jest ewaluacja strategii Obrońcy. W szczególnym przypadku, gdyby w populacji Atakującego były wszystkie możliwe strategie proste Atakującego, to działanie algorytmu CoEvoSG sprowadziłoby się do metody EASG, ponieważ znajdowane odpowiedzi Atakującego byłyby zawsze optymalne. Jednak, jak zostało zaznaczone na wstępie tego rozdziału, podejście koewolucyjne wprowadzono w celu ograniczenia liczby weryfikowanych strategii Atakującego i przyspieszenia dzia-

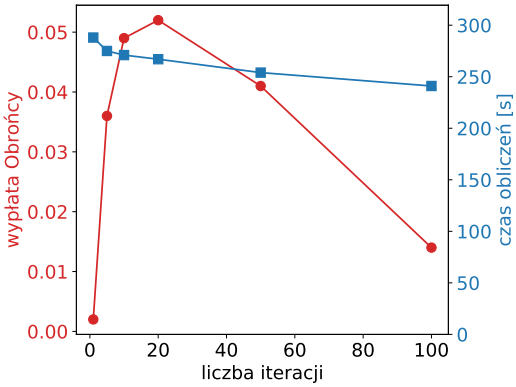
łania algorytmu. Na podstawie zademonstrowanych wyników przyjęto $N_A = 200$ jako rekomendowaną wartość wielkości populacji Atakującego.

Kolejnym testowanym parametrem była długość ewolucji każdej populacji mierzona liczbą pokoleń. Przypomnijmy, że w algorytmie CoEvoSG populacje Obrońcy i Atakującego są rozwijane naprzemiennie - najpierw jedna przez g_c pokoleń, potem druga przez g_c pokoleń, itd. Wielkość wypłaty Obrońcy oraz czas obliczeń w zależności od wartości parametru g_c zostały zaprezentowane na rysunku 9.2b. Można zaobserwować, że zarówno małe wartości ($g_c \leqslant 5$ - częste przełączanie między populacjami) jak i duże ($g_c \geqslant 50$) powodują obniżenie osiąganych wyników. Jednocześnie czas działania algorytmu we wszystkich przypadkach jest zbliżony. W dalszych eksperymentach przyjęto $g_c = 20$ jako rekomendowaną wartość.

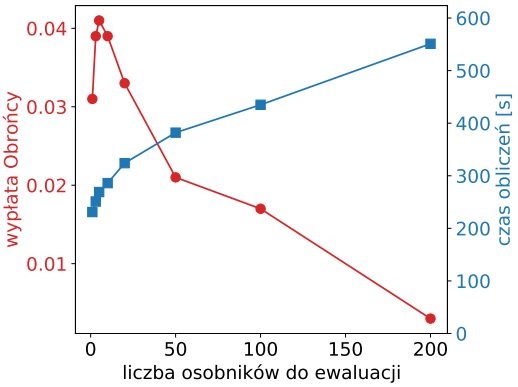
Ostatnim nowododanym parametrem w algorytmie koewolucyjnym jest liczba najlepszych osobników z populacji Obrońcy (N_{top}), względem których oceniane są strategie Atakującego. Wyniki przedstawione na rysunku 9.2c potwierdzają wcześniejsze spostrzeżenie dotyczące stosowania w tym celu całej populacji Obrońcy - $N_{top} = 200$. W takim przypadku widać znaczne obniżenie osiąganych rezultatów. Eksperymenty wykazały również, że zbyt małe wartości tego parametru $N_{top} < 5$ nie dają najlepszych wyników. Bliższa obserwacja zachowania populacji w takich przypadkach ujawniła tendencję algorytmu do oscylacji. Przykładowo, w przypadku, gdy strategia Atakującego jest oceniana względem tylko jednej najlepiej przystosowanej strategii Obrońcy ($N_{top} = 1$), to dość szybko populacja Atakującego traci różnorodność - wszystkie osobniki stają się podobne, bo optymalizują strategię jedynie względem tej jednej wybranej strategii Obrońcy. W efekcie populacja Atakującego potrafi jedynie zwracać dobrą odpowiedź na konkretną strategię Obrońcy, co powoduje, że w kolejnej fazie działania, populacja Obrońcy szybko znajduje inną strategię, dla której brakuje w populacji Atakującego dobrej odpowiedzi. W kolejnym kroku populacja Atakującego przystosowuje się do tej nowej strategii Obrońcy niejako "zapominając" o poprzednich. Testy wykazały, że wartość $N_{top} = 10$ zapewnia właściwy balans pomiędzy skrajnymi zachowaniami opisanymi powyżej.



(a) Wielkość populacji Atakującego (N_A).



(b) Liczba pokoleń między zmianą populacji (g_c).



(c) Liczba osobników biorących udział w ewaluacji strategii Atakującego (N_{top}).

Rysunek 9.2: Porównywanie wypłaty Obrońcy oraz czasu działania algorytmu w zależności od wielkości wybranych parametrów.

9.3.2 Gry testowe

Do testów użyto dwóch wcześniej opisanych rodzajów gier: Warehouse Games (WHG) oraz FlipIt Games (FIG). Gry SEG swoją charakterystyką przypominają WHG, a przestrzeń strategii Atakującego w grach SGP oraz SGS jest mała (jest to wybór jednego atakowanego wierzchołka), dlatego też nie zostały one uwzględnione w eksperymentach.

Aby w pełni zaprezentować potencjał algorytmu koewolucyjnego, przetestowane zostały również większe instancje gier, które znajdują się poza zasięgiem metod dotychczas opisanych w literaturze. Dla gier WHG przyjęto liczbę kroków czasowych $m \in \{3, 4, 5, 6, 8, 10, 15, 20\}$ oraz liczbę wierzchołków w grafie $|V| \in \{15, 20, 25, 30, 40, 50\}$. W przypadku gier FIG był to ten sam zbiór kroków czasowych oraz liczba wierzchołków $|V| \in \{5, 10, 15, 20, 25, 30, 40\}$. Dla każdej pary $(m, |V|)$ wygenerowano 5 gier z losowymi

wypłatami i połączeniami w grafie. W sumie, w eksperymentach sprawdzono 240 gier WHG i 280 gier FIG.

9.4 Wyniki

Tabele 9.1 i 9.2 przedstawiają porównanie średniej wartości wypłaty Obrońcy względem liczby kroków czasowych oraz wierzchołków w grafie. Porównywane metody zostały opisane w rozdziale 4. Myślniki oznaczają, że dana metoda nie była w stanie policzyć kompletu gier w zadanym limicie czasu, który wynosił 100h dla pojedynczej gry.

WHG				
$\|V\|$	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
15	0.052	0.051	0.051	0.050
20	0.054	0.053	0.052	0.050
25	0.048	0.046	0.045	0.043
30	-	0.044	0.042	0.039
40	-	-	0.040	0.036
50	-	-	-	0.029

FIG				
$\|V\|$	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
5	0.890	0.887	0.886	0.886
10	0.854	0.851	0.847	0.845
15	0.811	0.807	0.802	0.798
20	-	0.784	0.780	0.772
25	-	-	0.754	0.746
30	-	-	-	0.730
40	-	-	-	0.722

Tabela 9.1: Porównanie średnich oczekiwanych wypłat Obrońcy w zależności od liczby wierzchołków w grafie. Wyniki są uśrednione po wszystkich wartościach m.

WHG				
m	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
3	0.043	0.043	0.043	0.043
4	0.052	0.050	0.050	0.049
5	0.055	0.054	0.053	0.052
6	0.058	0.056	0.054	0.051
8	-	0.053	0.051	0.048
10	-	-	0.048	0.044
15	-	-	-	0.040
20	-	-	-	0.038

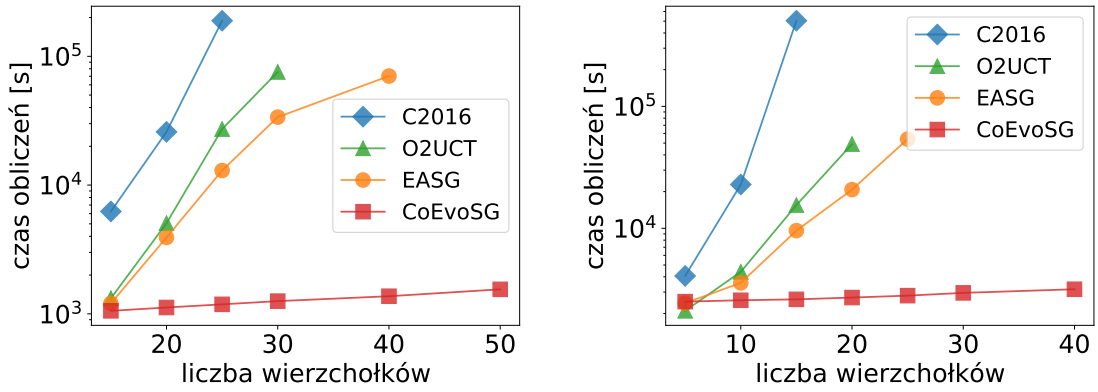
FIG				
m	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
3	0.823	0.821	0.820	0.817
4	0.817	0.812	0.808	0.805
5	0.810	0.801	0.798	0.791
6	-	0.794	0.792	0.791
8	-	0.789	0.784	0.781
10	-	-	0.780	0.778
15	-	-	-	0.774
20	-	-	-	0.761

Tabela 9.2: Porównanie średnich oczekiwanych wypłat Obrońcy w zależności od liczby kroków czasowych. Wyniki są uśrednione po wszystkich wartościach $|V|$.

Przedstawione w tabeli dane pokazują, że różnica między wynikami osiągniętymi przez algorytm EASG a rozszerzoną wersję koewolucyjną (CoEvoSG) jest nieduża. W przypadku gier WHG wyniosła ona średnio 0.0020, a dla FIG 0.0032. Ta niewielka różnica wynika z faktu, że w większości przypadków (84%) populacja Atakującego w CoEvoSG zawierała optymalną odpowiedź Atakującego na najlepszą strategię z populacji Obrońcy, więc wynik ewaluacji był taki sam jak w metodzie EASG.

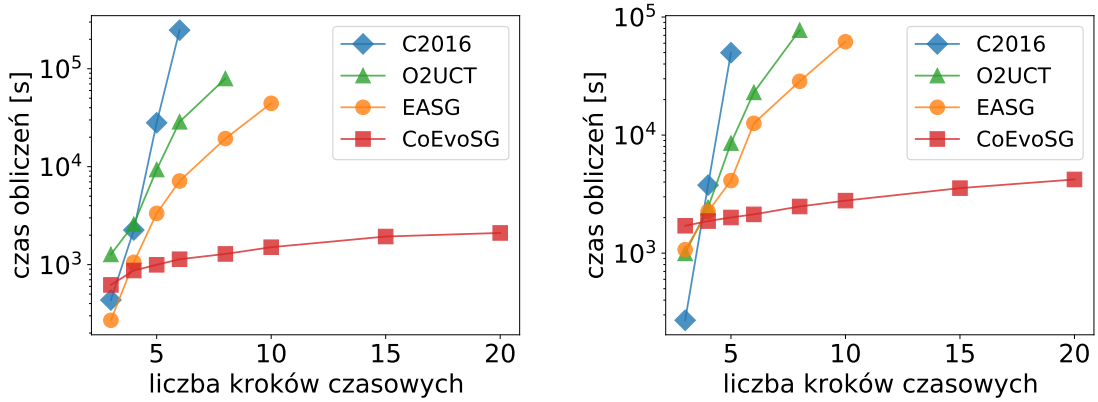
Algorytm dokładny C2016 był w stanie, w zadanym limicie czasu, policzyć wyniki (dokładne) dla 60 gier WHG i 45 gier FIG. Spośród nich CoEvoSG zwrócił optymalny wynik odpowiednio dla 38/60 (68%) oraz 29/45 (64%) przypadków. Średnia różnica między rezultatami optymalnymi a zwracanymi przez CoEvoSG wyniosła 0.0023 (WHG) oraz 0.0137 (FIG).

Na rysunkach 9.3 oraz 9.4 przedstawiono czas obliczeń porównywanych metod w zależności od liczby wierzchołków i liczby kroków czasowych. We wszystkich przypadkach dostrzec można wyraźną przewagę algorytmu koewolucyjnego. Wykazuje on niemal stały czas obliczeń niezależnie od wielkości gry, podczas gdy inne metody skalują się w przybliżeniu liniowo (O2UCT i EASG) lub wykładniczo (C2016).



(a) WHG (b) FIG

Rysunek 9.3: Porównywanie czasu obliczeń (skala logarytmiczna) w zależności od liczby wierzchołków ($|V|$). Wyniki są uśrednione po wszystkich wartościach m.



(a) WHG (b) FIG

Rysunek 9.4: Porównywanie czasu obliczeń (skala logarytmiczna) w zależności od liczby kroków czasowych (m). Wyniki są uśrednione po wszystkich wartościach $|V|$.

Porównując czas obliczeń z otrzymywanymi wypłatami prezentowanymi w tabelach powyżej, można zauważyć następującą zależność: im lepsze wyniki zwracane przez daną metodę, tym dłuższy jej czas działania, gorsza skalowalność i brak możliwości policzenia większych gier w zadanym limicie czasu.

Zaprezentowane wyniki pokazały, że podejście koewolucyjne, mimo osiągania nieznacznie gorszych rezultatów niż podstawowa metoda EASG, może być użyteczną alternatywą w przypadku większych gier dzięki dużo lepszej skalowalności czasowej.

Rozdział 10

Podsumowanie

10.1 Wady i zalety proponowanych rozwiązań

W rozprawie zaprezentowano algorytm ewolucyjny (EASG) do poszukiwania równowagi w grach obronnych Stackelberga. Następnie przedstawiony został szereg jego adaptacji do różnych typów gier, np. gry na płaszczyźnie (EASG_{SGP}), gry z sygnalizacją (EASG_{SGS}), gry z częściową obserwowalnością (EASG_{SEG}), czy gry z niepełną racjonalnością Atakującego. Dodatkowo zaproponowane zostały podejścia neuroewolucyjne (NESG) oraz koewolucyjne (CoEvoSG).

Wyniki przeprowadzonych eksperymentów oraz analizy prezentowanych metod pozwalają na wyciągnięcie wniosków dotyczących ich silnych i słabych stron. Wśród najbardziej istotnych zalet można wymienić:

- **Dobra skalowalność czasowa.** W przeważającej większości eksperymentów czas obliczeń proponowanych algorytmów był zdecydowanie niższy niż metod dotychczas opisywanych w literaturze. Zaproponowane w rozprawie podejścia charakteryzują się lepszą wydajnością obliczeniową, co pozwala na rozwiązywanie gier większych i bardziej skomplikowanych, będących do tej pory poza zasięgiem konkurencyjnych metod.

- **Wyniki bliskie optymalnym.** Przeprowadzone eksperymenty wykazały, że w przeważającej części przypadków strategie Obrońcy zwracane przez przedstawione algorytmy ewolucyjne były optymalne lub bardzo bliskie optymalnym, co w większości zastosowań praktycznych jest wystarczające.

- **Małe wymagania pamięciowe.** W porównaniu do innych metod zaproponowane algorytmy zużywają znacznie mniej zasobów pamięciowych. Szczególnie w przypadku metod opartych o programowanie liniowe, nadmierne wykorzystanie pamięci często uniemożliwia rozwiązywanie większych gier o bardziej skomplikowanej strukturze. Zaproponowane algorytmy wykazują w przybliżeniu stałe zużycie pamięci niezależnie od wielkości gry i poziomu jej skomplikowania.

- **Uniwersalność.** Zaproponowane podejście pozwala na łatwą adaptację do różnych rodzajów gier i ich wariantów. W eksperymentach rozważanych było kilka rodzajów gier o odmiennej charakterystyce: z jednym lub wieloma zasobami Obrońcy, rozgrywane na grafie lub w przestrzeni ciągłej, z częściową lub pełną obserwowalnością, z pełną lub ograniczoną racjonalnością, czy z uwzględnioną niepewnością detekcji. We wszystkich tych przypadkach zaproponowana bazowa wersja algorytmu ewolucyjnego była możliwa do zastosowania i nie wymagała istotnych modyfikacji czy specjalnego doboru wartości parametrów.

- **Charakterystyka Anytime.** Działanie zaproponowanych algorytmów może być przerwane w dowolnym momencie, co skutkuje zwróceniem najlepszego dotychczas znalezionego rozwiązania. Taka własność (ang. *anytime characterstics*) jest szczególnie istotna w systemach, które muszą działać w restrykcyjnie określonym limicie czasu lub nie można z góry przewidzieć jaki ten limit będzie i ich działanie może zostać przerwane w trakcie. W takich sytuacjach nie można zastosować metod, które budują rozwiązanie stopniowo i dopóki nie zostanie ono w pełni zbudowane, to jest bezużyteczne z praktycznego punktu widzenia. Proponowane algorytmy ewolucyjne od początku operują na kompletnych rozwiązaniach, a w kolejnych pokoleniach poprawiana jest ich jakość, co umożliwia otrzymanie poprawnego rezultatu także w sytuacji zatrzymania obliczeń w dowolnym momencie.

Proponowane rozwiązania również nie są pozbawione wad. Do ich słabych stron należą:

- **Brak teoretycznej gwarancji jakości osiąganych rezultatów.** Pomimo że przedstawione wyniki eksperymentów wskazują, że zaproponowane algorytmy w zdecydowanej większości przypadków zwracają rozwiązania bliskie optymalnym lub optymalne, to nie ma teoretycznej gwarancji, że tak będzie dla wszystkich gier. Co jest charakterystyczne dla metod ewolucyjnych (czy w ostatnich latach również dla większości metod inteligencji obliczeniowej, w tym np. sieci neuronowych), brakuje matematycznego dowodu czy formalnych gwarancji zapewnienia określonego poziomu jakości rozwiązań w ogólnym przypadku.

- **Mnogość parametrów.** Proponowane rozwiązania zawierają szereg parametrów takich jak wielkość populacji, limit pokoleń, prawdopodobieństwo mutacji/krzyżowania, presja selekcji, itd. Z jednej strony, dzięki temu algorytmy mogą być łatwo dostosowane do różnych potrzeb, lecz z drugiej strony, strojenie tych parametrów może być czasochłonne i uciążliwe. Na szczęście przeprowadzone eksperymenty wykazały brak istotnej wrażliwości algorytmów (zmiany otrzymywanych wyników) na manipulację poszczególnymi parametrami, a te same rekomendowane wartości były odpowiednie dla różnych rodzajów gier.

10.2 Odniesienie do hipotezy badawczej

Zaproponowane w rozprawie algorytmy ewolucyjne, ich analiza oraz wyniki przeprowadzonych eksperymentów pozwalają stwierdzić, że metody ewolucyjne z powodzeniem mogą być wykorzystywane w poszukiwaniu równowagi w grach obronnych Stackelberga. Zwracane przez nie rezultaty są bliskie optymalnym i porównywalne z wynikami innych znanych w literaturze algorytmów przybliżonych. Przedstawione podejścia ewolucyjne charakteryzują się najlepszą skalowalnością czasu obliczeń oraz zużycia pamięci spośród wszystkich porównywanych metod, co pozwala na rozwiązywanie większych i bardziej skomplikowanych instancji gier, które są poza zasięgiem konkurencyjnych algorytmów. Jednocześnie zaproponowane podejście dzięki ogólności sformułowania pozwala na ła-

twą adaptację i szerokie zastosowanie do gier o różnej charakterystyce (np. z ograniczoną racjonalnością, częściową obserwowalnością, sygnalizacją). Według najlepszej wiedzy autora, zaprezentowane w tej rozprawie podejście ewolucyjne jest pierwszym w literaturze zastosowaniem algorytmów ewolucyjnych do rozwiązywania wielokrokowych gier obronnych Stackelberga. Przedstawione modyfikacje i rozszerzenia metody z pewnością nie wyczerpują potencjału zastosowania metod ewolucyjnych w omawianym obszarze, jednak stanowią potwierdzenie ich użyteczności i mogą stanowić punkt wyjścia do dalszych, bardziej szczegółowych badań lub wdrożeń praktycznych.

Na podstawie zaprezentowanych wyników można stwierdzić, że postawiona na wstępie rozprawy **hipoteza badawcza, stwierdzająca możliwość zastosowania algorytmów ewolucyjnych do efektywnej aproksymacji strategii w stanie równowagi w grach obronnych Stackelberga, jest prawdziwa**.

10.3 Możliwości dalszego rozwoju

Mimo przeprowadzenia obszernej analizy zaproponowanego algorytmu ewolucyjnego oraz jego wariantów, nadal istnieje kilka obszarów badawczych, które mogą zostać rozwinięte w przyszłości. Są to między innymi:

- **Testy na rzeczywistych danych.** Analizowane zagadnienie gier obronnych ściśle związane jest z dziedziną bezpieczeństwa. Firmy czy instytucje państwowe pracujące w tych obszarach niechętnie dzielą się wiedzą i danymi. Najczęściej powodem są tajemnice przedsiębiorstwa, klientów lub konieczność zapewnienia poufności, gdyż upublicznienie takich danych mogłoby spowodować niebezpieczeństwo, np. łatwiejsze zaplanowanie ataku. Dlatego też przedstawione w rozprawie eksperymenty były przeprowadzone na danych sztucznie wygenerowanych przez autora lub pozyskanych z literatury. Wartym uwagi eksperymentem byłaby weryfikacja zaproponowanego rozwiązania na rzeczywistych danych pozyskanych z instytucji zajmujących się tematyką bezpieczeństwa.

- **Eksperymenty z udziałem ludzi.** W rozdziale 8 zwracano szczególną uwagę na ograniczenia pojawiające się podczas rozgrywek z udziałem ludzi. Przedstawione

teorie ograniczonej racjonalności oparte są na pracach i badaniach psychologów. Jednak interesującym aspektem byłoby przeprowadzenie takich eksperymentów z udziałem ludzi ściśle w kontekście gier obronnych Stackelberga, co pozwoliłoby na weryfikację poczynionych założeń i skuteczności zaproponowanych rozwiązań.

- **Inne warianty gier.** W rozprawie uwaga skupiona była na grach sekwencyjnych. Jednak jest wiele innych rodzajów gier obronnych, do rozwiązywania których można byłoby użyć metody zaproponowanej w rozprawie, po jej odpowiedniej modyfikacji. W wielu przypadkach adaptacji wymagałyby tylko niektóre elementy algorytmu. Przykładami wariantów gier, które mogłyby zostać przetestowane, są gry z wieloma Obrońcami [43, 50] i/lub wieloma Atakującymi [40], gry z niekompletną informacją [53], czy ostatnio sformułowane gry sekwencyjne z pamięcią [3].

- **Praktyczne wdrożenie systemu.** Przeprowadzone w rozprawie eksperymenty pokazały, że omawiany algorytm cechuje się dobrą skalowalnością czasową, niewielkimi wymaganiami pamięciowymi oraz zdolnością do powtarzalnego osiągania wyników bliskich optymalnym. Z tego względu może być dobrym kandydatem do praktycznego wdrożenia. Ze względu na łatwość adaptacji do różnych gier (z różnymi zasadami i o różnej charakterystyce) oraz możliwość przystępnej parametryzacji, może on stanowić wartościową alternatywę dla systemów używanych obecnie. Próba znalezienia możliwości praktycznego wdrożenia zaproponowanego rozwiązania stanowić będzie cel przyszłych prac w tym obszarze.

Bibliografia

[1] Bo An, Fernando Ordóñez, Milind Tambe, Eric Shieh, Rong Yang, Craig Baldwin, Joseph DiRenzo III, Kathryn Moretti, Ben Maule, and Garrett Meyer. A deployed quantal response-based patrol planning system for the US Coast Guard. *Interfaces*, 43(5):400–420, 2013.

[2] Jarosław Arabas. *Wykłady z algorytmów ewolucyjnych*. Wydawnictwa Naukowo-Techniczne, 2004.

[3] Aditya Aradhye, Branislav Bošanskỳ, and Michael Hlaváček. Computing Stackelberg Equilibrium with memory in Sequential Games. *arXiv preprint arXiv:2111.02111*, 2021.

[4] Tobias Blickle and Lothar Thiele. A mathematical analysis of tournament selection. W *ICGA*, wolumen 95, strony 9–15. Citeseer, 1995.

[5] Elizabeth Bondi, Hoon Oh, Haifeng Xu, Fei Fang, Bistra Dilkina, and Milind Tambe. To signal or not to signal: Exploiting uncertain real-time information in signaling games for security and sustainability. W *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, wolumen 34, strony 1369–1377, 2020.

[6] Branislav Bošanskỳ, Simina Brânzei, Kristoffer Arnsfelt Hansen, Troels Bjerre Lund, and Peter Bro Miltersen. Computation of Stackelberg Equilibria of finite sequential games. *ACM Transactions on Economics and Computation*, 5(4):1–24, 2017.

[7] Branislav Bošanskỳ, Simina Brânzei, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. Computation of Stackelberg Equilibria of finite sequential games. *Lecture Notes in Computer Science*, strony 201–215, 2015.

[8] Branislav Bošanskỳ and Jiri Cermak. Sequence-form algorithm for computing Stackelberg Equilibria in extensive-form games. W Blai Bonet and Sven Koenig, editors, *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, strony 805–811. AAAI Press, 2015.

[9] Branislav Bošanskỳ and Jiri Cermak. Sequence-form algorithm for computing Stackelberg Equilibria in extensive-form games. W *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[10] Michele Breton, Abderrahmane Alj, and Alain Haurie. Sequential Stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1):71–97, 1988.

[11] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1–43, 2012.

[12] Victor Bucarey, Carlos Casorrán, Óscar Figueroa, Karla Rosas, Hugo Navarrete, and Fernando Ordóñez. Building real Stackelberg Security Games for border patrols. W *Proceedings of the International Conference on Decision and Game Theory for Security*, strony 193–212. Springer, 2017.

[13] Jiri Cermak, Branislav Bošanskỳ, Karel Durkota, Viliam Lisy, and Christopher Kiekintveld. Using correlated strategies for computing Stackelberg Equilibria in extensive-form games. W *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, strony 439–445. AAAI Press, 2016.

[14] Jakub Černỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. Incremental strategy generation for Stackelberg Equilibria in extensive-form games. W *Proceedings of the ACM Conference on Economics and Computation*, strony 151–168, 2018.

[15] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11-12):1245–1287, 2002.

[16] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[17] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. W *Proceedings of the 7th ACM conference on Electronic commerce*, strony 82–90. ACM, 2006.

[18] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

[19] Kahneman Daniel. Thinking, fast and slow, 2011.

[20] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[21] Richard Dawkins and Nicola Davis. *The Selfish Gene*. Macat Library, 2017.

[22] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M Bahaa-Eldin. A survey on deep packet inspection. W *Proceedings of the 12th International Conference on Computer Engineering and Systems*, strony 188–197. IEEE, 2017.

[23] Fei Fang, Shutian Liu, Anjon Basak, Quanyan Zhu, Christopher D Kiekintveld, and Charles A Kamhoua. Introduction to game theory. *Game Theory and Machine Learning for Cyber Security*, strony 21–46, 2021.

[24] Fei Fang and Thanh H Nguyen. Green security games: Apply game theory to addressing green security challenges. *ACM SIGecom Exchanges*, 15(1):78–83, 2016.

[25] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying PAWS: Field optimization of the protection assistant for wildlife security. W *Proceedings of the 28th Innovative Applications of Artificial Intelligence Conference*, 2016.

[26] Fei Fang, Peter Stone, and Milind Tambe. When Security Games go green: Designing defender strategies to prevent poaching and illegal fishing. W *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2015.

[27] Sevan Gregory Ficici. *Solution Concepts in Coevolutionary Algorithms.* PhD thesis, Brandeis University, 2004.

[28] William Haskell, Debarun Kar, Fei Fang, Milind Tambe, Sam Cheung, and Elizabeth Denicola. Robust protection of fisheries with compass. W *Proceedings of the 26th Innovative Applications of Artificial Intelligence Conference*, 2014.

[29] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. W *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, wolumen 24, 2010.

[30] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the LAX airport police and the Federal Air Marshal Service. *Interfaces*, 40(4):267–290, 2010.

[31] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. W *Handbook of the fundamentals of financial decision making: Part I*, strony 99–127. World Scientific, 2013.

[32] Debarun Kar, Fei Fang, Francesco Delle Fave, Nicole Sintov, and Milind Tambe. A Game of Thrones : When human behavior models compete in repeated Stackelberg Security Games. *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, strony 1381–1390, 2015.

[33] Jan Karwowski and Jacek Mańdziuk. A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research*, 2019.

[34] Jan Karwowski and Jacek Mańdziuk. Stackelberg equilibrium approximation in general-sum extensive-form games with Double-Oracle Sampling method. W *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, strony 2045–2047, 2019.

[35] Jan Karwowski and Jacek Mańdziuk. Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games. W *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, wolumen 34, strony 2054–2061, 2020.

[36] Jan Karwowski, Jacek Mańdziuk, and Adam Żychowski. Anchoring Theory in Sequential Stackelberg Games. W *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, strony 1881–1883, 2020.

[37] Jan Karwowski, Jacek Mańdziuk, Adam Żychowski, Filip Grajek, and Bo An. A memetic approach for sequential Security Games on a plane with moving targets. W *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, wolumen 33, strony 970–977, 2019.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. W *European Conference on Machine Learning*, strony 282–293. Springer, 2006.

[40] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Security games with multiple attacker resources. W *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.

[41] Nupul Kukreja, William GJ Halfond, and Milind Tambe. Randomizing regression tests using game theory. W *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, strony 616–621. IEEE, 2013.

[42] George Leitmann. On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4):637–643, Dec 1978.

[43] Jian Lou, Andrew M Smith, and Yevgeniy Vorobeychik. Multidefender security games. *IEEE Intelligent Systems*, 32(1):50–60, 2017.

[44] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor

Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[45] Jacek Mańdziuk, Jan Karwowski, and Adam Żychowski. Simulation-based methods in multi-step Stackelberg Security Games in the context of homeland security, 2022.

[46] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38, 1995.

[47] Zbigniew Michalewicz. *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Wydawnictwa Naukowo-Techniczne, 2003.

[48] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[49] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[50] Dolev Mutzari, Jiarui Gan, and Sarit Kraus. Coalition formation in multi-defender Security Games. W *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, strony 5603–5610, 2021.

[51] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. W *Proceedings of the 14th International Conference on Machine Learning*, 2010.

[52] Cong T Nguyen, Diep N Nguyen, Dinh Thai Hoang, Hoang-Anh Pham, Nguyen Huynh Tuong, and Eryk Dutkiewicz. Blockchain and Stackelberg game model for roaming fraud prevention and profit maximization. W *IEEE Wireless Communications and Networking Conference*, strony 1–6. IEEE, 2020.

[53] Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Security games with incomplete information. W *Proceedings of the IEEE International Conference on Communications*, strony 1–6. IEEE, 2009.

[54] Thanh Nguyen, Michael P Wellman, and Satinder Singh. A Stackelberg game model for botnet data exfiltration. W *Proceedings of the International Conference on Decision and Game Theory for Security*, strony 151–170. Springer, 2017.

[55] Thanh Hong Nguyen, Debarun Kar, Matthew Brown, Arunesh Sinha, Albert Xin Jiang, and Milind Tambe. Towards a science of security games. W *Mathematical sciences with multidisciplinary applications*, strony 347–381. Springer, 2016.

[56] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. W *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, strony 718–724, 2013.

[57] Swetasudha Panda and Yevgeniy Vorobeychik. Stackelberg games for vaccine design. W *14th International Conference on Autonomous Agents and Multiagent Systems*, strony 1391–1399, 2015.

[58] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Efficient algorithms to solve Bayesian Stackelberg Games for security applications. W *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, strony 1559–1562, 2008.

[59] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. W *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, wolumen 2, strony 895–902, 2008.

[60] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. W *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, strony 125–132, 2008.

[61] James Pita, Manish Jain, Fernando Ordóñez, Milind Tambe, Sarit Kraus, Reuma Magori-Cohen, and Milind Tambe. Effective solutions for real-world Stackelberg Games: When agents must deal with human uncertainties. *Security and Game Theory*, strony 193–212, 2011.

[62] James Pita, Richard John, Rajiv Maheswaran, Milind Tambe, Rong Yang, and Sarit Kraus. A robust approach to addressing human adversaries in security games. W *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, strony 1297–1298, 2012.

[63] Tadeusz Płatkowski. Wstęp do teorii gier. *Uniwersytet Warszawski*, 2012.

[64] Elena Popovici, Anthony Bucci, R Paul Wiegand, and Edwin D De Jong. Coevolutionary principles, 2012.

[65] Klaus Ritzberger et al. *The theory of extensive form games*. Springer, 2016.

[66] Ariel Rosenfeld and Sarit Kraus. When Security Games hit traffic: Optimal traffic enforcement under one sided uncertainty. W *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, strony 3814–3822, 2017.

[67] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT: A deployed game theoretic system to protect the ports of the United States. W *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, strony 13–20, 2012.

[68] Herbert A Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[69] Herbert A Simon. *Models of Man: Social and Rational*. Wiley, 1957.

[70] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg Security Games: looking beyond a decade of success. W *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, strony 5494–5501, 7 2018.

[71] Arunesh Sinha, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, and Albert Xin Jiang. From physical security to cybersecurity. *Journal of Cybersecurity*, 1(1):19–35, 2015.

[72] Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. IRIS-a tool for strategic security allocation in transportation networks. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, strony 37–44, 2009.

[73] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[74] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, 5(4):297–323, 1992.

[75] Marten Van Dijk, Ari Juels, Alina Oprea, and Ronald L Rivest. FlipIt: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.

[76] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. W *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, wolumen 30, 2016.

[77] Aravind Venugopal, Elizabeth Bondi, Harshavardhan Kamarthi, Keval Dholakia, Balaraman Ravindran, and Milind Tambe. Reinforcement learning for unified allocation and patrolling in signaling games with uncertainty. W *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, strony 1353–1361, 2021.

[78] Bernhard Von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Technical report, CDAM Research Report, 2004.

[79] Xinrun Wang, Bo An, Martin Strobel, and Fookwai Kong. Catching captain jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. W *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, wolumen 32, 2018.

[80] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. Deep reinforcement learning for Green Security Games with real-time information. W *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

[81] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[82] Karsten Weicker. *Evolutionary algorithms and dynamic optimization problems*. Der Andere Verlag Berlin, 2003.

[83] Haifeng Xu, Zinovi Rabinovich, Shaddin Dughmi, and Milind Tambe. Exploring information asymmetry in two-stage security games. W *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, wolumen 29, 2015.

[84] Rong Yang, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Richard John. Improving resource allocation strategy against human adversaries in security games. W *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, strony 458–464, 2011.

[85] Rong Yang, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, and Richard John. Improving resource allocation strategies against human adversaries in Security Games: An extended study. *Artificial Intelligence*, 195(195):440–469, 2013.

[86] Zhengyu Yin, Albert Xin Jiang, Matthew P Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. W *Proceedings of the 24th Innovative Applications of Artificial Intelligence Conference*, 2012.

[87] Adam Żychowski and Jacek Mańdziuk. A generic metaheuristic approach to sequential Security Games. W *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, strony 2089–2091, 2020.

[88] Adam Żychowski and Jacek Mańdziuk. Evolution of strategies in sequential Security Games. W *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, strony 1434–1442, 2021.

[89] Adam Żychowski and Jacek Mańdziuk. Learning attacker's bounded rationality model in security games. W *Proceedings of the 19th International Conference on Neural Information Processing*, strony 530–539. Springer, 2021.

[90] Adam Żychowski and Jacek Mańdziuk. Coevolutionary approach to sequential Stackelberg Security Games. W *International Conference on Computational Science*, 2022 (przyjęta do druku).

[91] Adam Żychowski, Jacek Mańdziuk, Elizabeth Bondi, Aravind Venugopal, Milind Tambe, and Balaraman Ravindran. Evolutionary approach to Security Games with signaling. W *International Joint Conference on Artificial Intelligence*, 2022 (przyjęta do druku).